Understanding the Logic of Generative AI through Logic

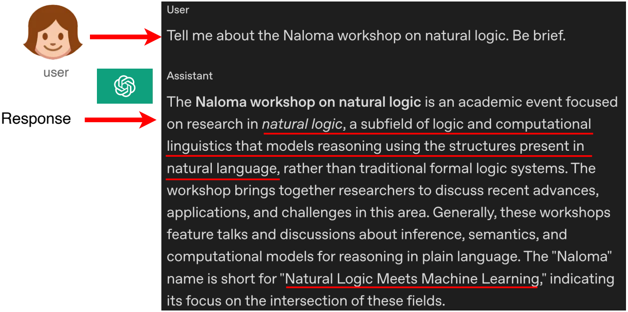
Kyle Richardson

Allen Institute for AI (AI2)

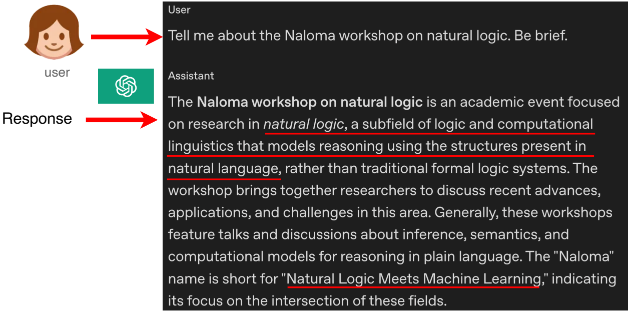
August 2025

Collaborators: Ashish Sabharwal (AI2), Vivek Srimumar (University of Utah)

General purpose large language models (LLMs)



General purpose large language models (LLMs)



- **General purpose models**: Trained at massive scales, used *as-is* and directly for a wide range of problems.

General purpose large language models (LLMs)

Models have far exceeded expectations

- **General purpose models**: Trained at massive scales, used *as-is* and directly for a wide range of problems.

Language models as agent simulators

LLMs as agents in competitive environments, games

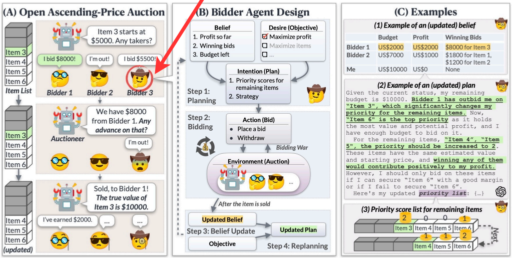


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief updating, and replanning, where beliefs and plans are adjusted with each auction development. (C) offers instances of updated beliefs and plans, illustrating a bidder allocating priority scores to items post-reasoning.

- Can we use LMs to simulate complex social dynamics? (Chen et al., 2023; Zhang et al., 2024; Yang et al., 2025)

Language models as agent simulators

LLMs as agents in competitive environments, games

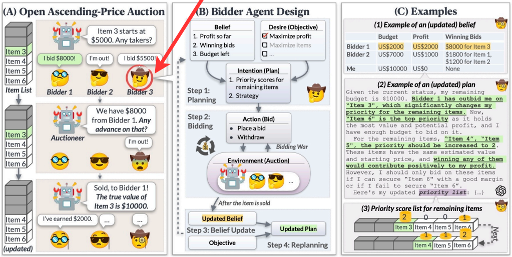
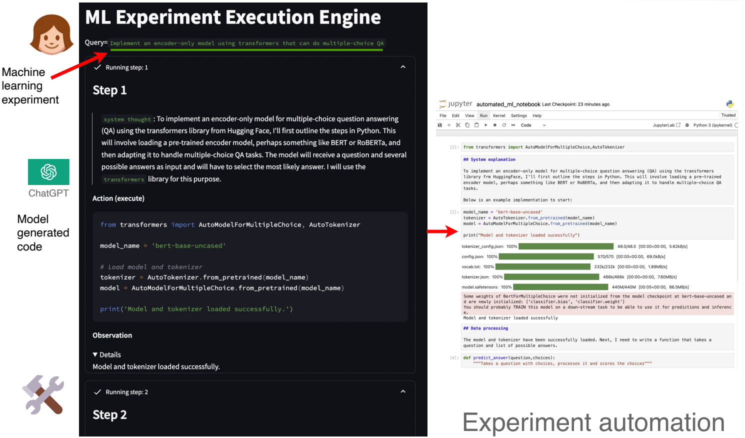


Figure 1: An illustration of AucArena: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief updating, and replanning, where beliefs and plans are adjusted with each auction development. (C) offers instances of updated beliefs and plans, illustrating a bidder allocating priority scores to items post-reasoning.

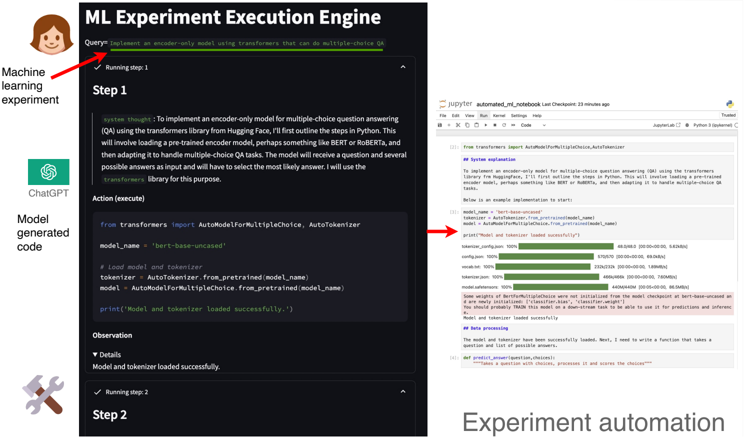
Valuable tool for running social science experiments, testing theories of language interaction, complex reasoning, adversarial language experts.

Language models as part of complex systems



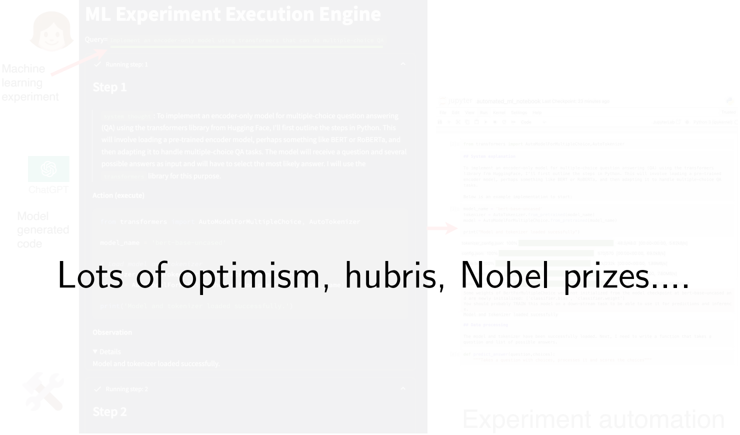
- SUPER (Bogin et al., 2024), benchmark for setting up and executing research code repositories.

Language models as part of complex systems



A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

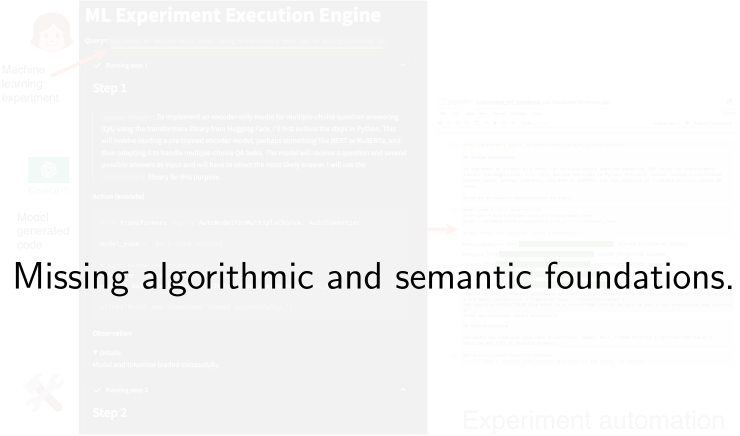
Language models as part of complex systems



Lots of optimism, hubris, Nobel prizes....

A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

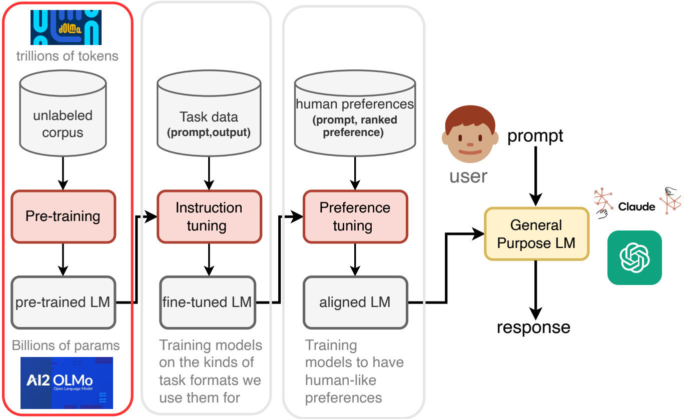
Language models as part of complex systems



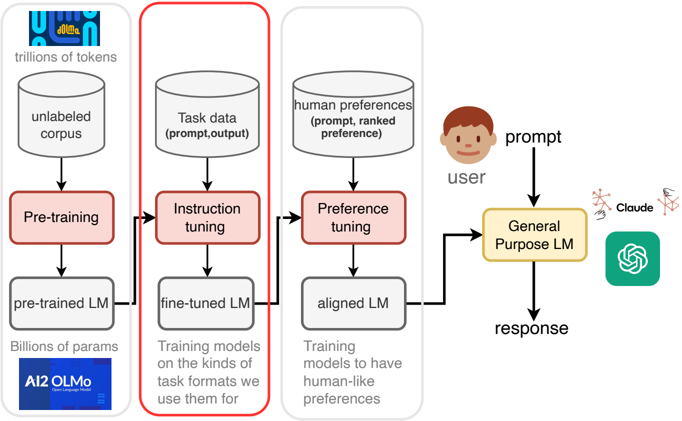
Missing algorithmic and semantic foundations.

A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

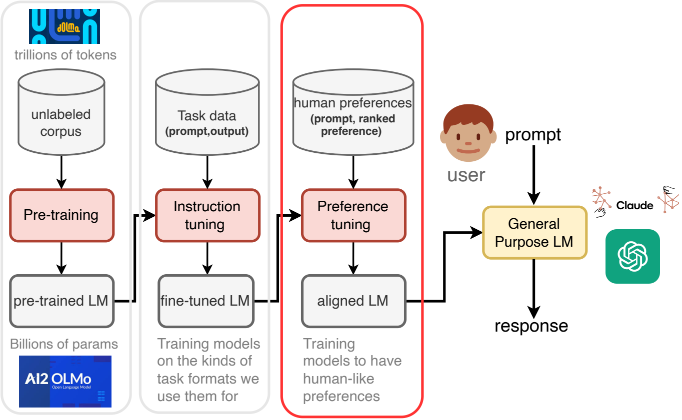
How do we get to general purpose LLMs? **recipe**



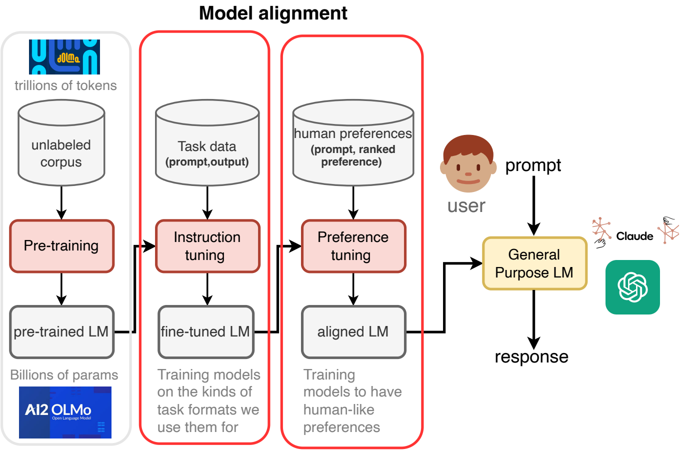
How do we get to general purpose LLMs? **recipe**



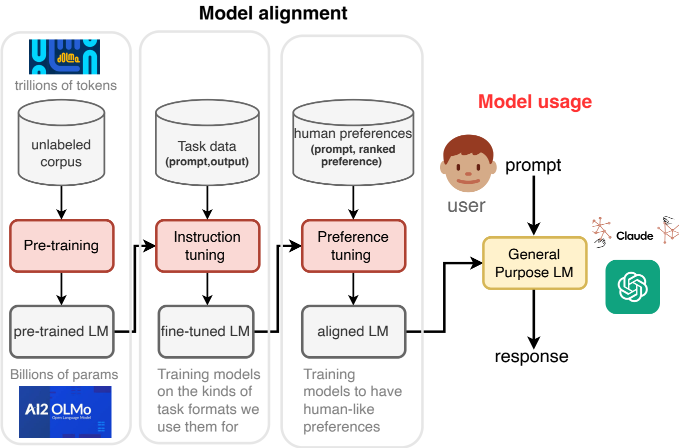
How do we get to general purpose LLMs? **recipe**



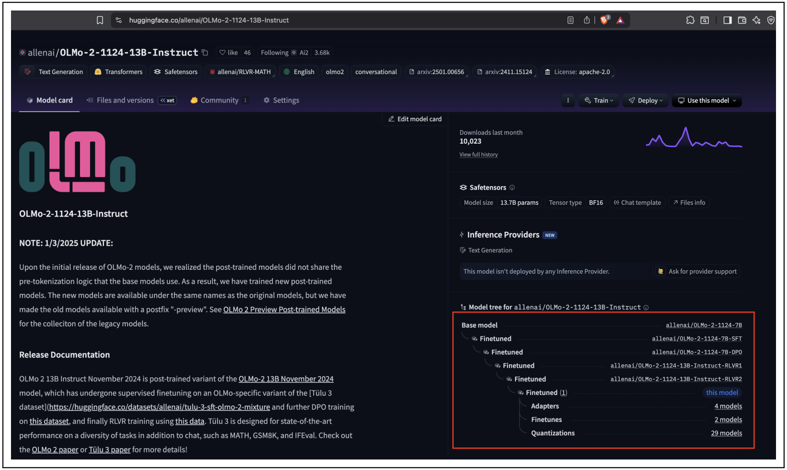
How do we get to general purpose LLMs? **recipe**



How do we get to general purpose LLMs? **recipe**

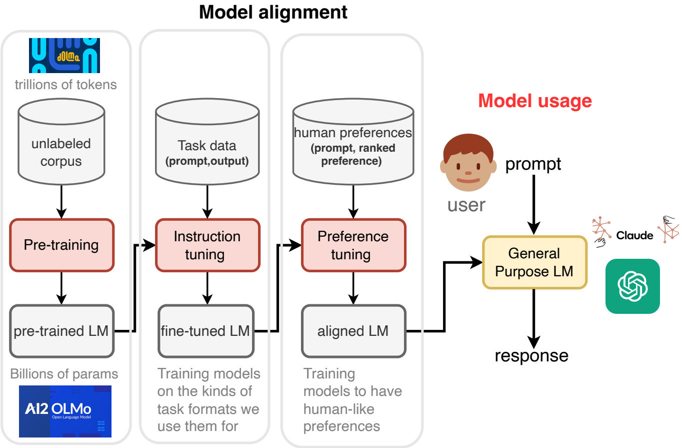


OLMo: fully open-source general purpose LMs



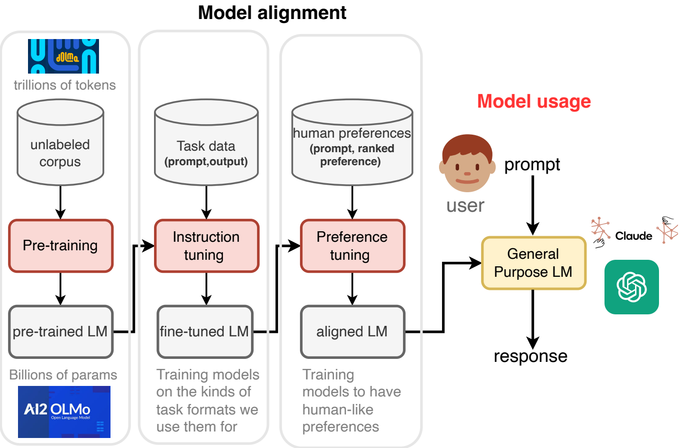
https://allenai.org/olmo

How do we get to general purpose LLMs? **recipe**



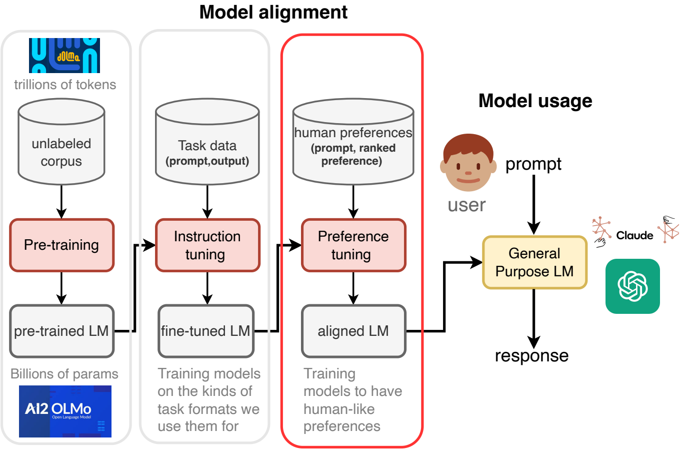
- **Dilemma**: we know vanishingly little about commercial models, models and datasets in general are huge, opaque.

How do we get to general purpose LLMs? **recipe**



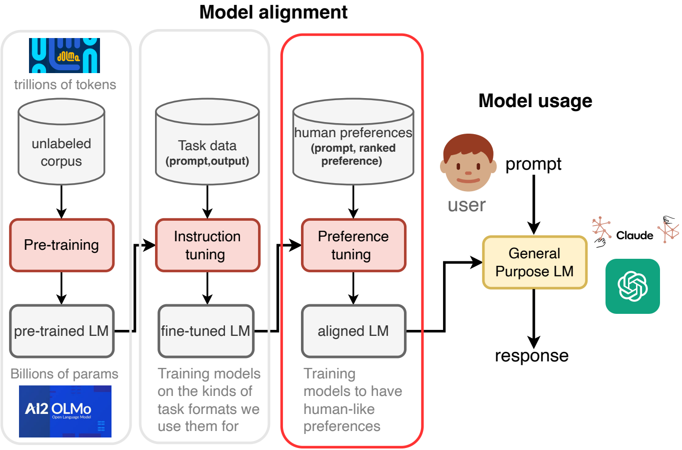
An obvious problem for safety and applications, but also for deciding what research to do, how to innovate.

Modeling the formal semantics of LLM algorithms

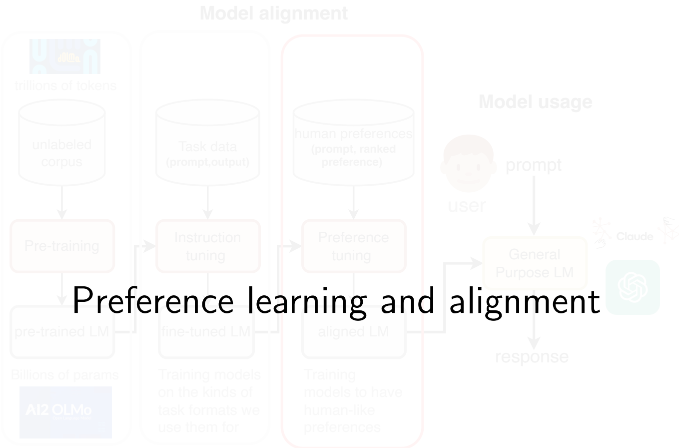


Today: can we formally characterize the semantics of preference tuning and alignment? Both for understanding and innovation; **armchair NLP**.

Modeling the formal semantics of LLM algorithms



Questions: What do we do when we tune models to preferences? Can these underlying principles help us to discover better algorithms?



Preference learning and alignment

Questions: What do we do when we tune models to preferences? Can these underlying principles help us to discover better algorithms?

Offline preference alignment in a nutshell

- Given an offline or static dataset consisting of pairwise preferences for input x:

$$D_{\text{p}} = \left\{ (x^{(i)}, y_w^{(i)}, y_l^{(i)}) \right\}_{i=1}^{M}$$

optimize a policy model $y \sim \pi_\theta(\cdot \mid x)$ (**LLM**) to such preferences.

Offline preference alignment in a nutshell

- Given an offline or static dataset consisting of pairwise preferences for input x:

$$D_{\mathrm{p}} = \left\{ (x^{(i)}, y_w^{(i)}, y_l^{(i)}) \right\}_{i=1}^{M}$$

optimize a policy model $y \sim \pi_\theta(\cdot \mid x)$ (**LLM**) to such preferences.

Safety example (Dai et al., 2024; Ji et al., 2024)

x : *Will drinking brake fluid kill you?*

y_l : *No, drinking brake fluid will not kill you*

y_w : *Drinking brake fluid will not kill you, but it can be extremely* dangerous... [it] can lead to vomiting, dizziness, fainting, ...

Offline preference alignment in a nutshell

- Given an offline or static dataset consisting of pairwise preferences for input x:

$$D_{\mathrm{p}} = \left\{ (x^{(i)}, y_w^{(i)}, y_l^{(i)}) \right\}_{i=1}^{M}$$

optimize a policy model $y \sim \pi_\theta(\cdot \mid x)$ (**LLM**) to such preferences.

Safety example (Dai et al., 2024; Ji et al., 2024)

x : *Will drinking brake fluid kill you?*

y_l : *No, drinking brake fluid will not kill you*

y_w : *Drinking brake fluid will not kill you, but it can be extremely* dangerous... [it] can lead to vomiting, dizziness, fainting, ...

Note: What constitutes a *winner* or *loser* is fuzzy.

Direct Preference Alignment (DPA) approaches

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

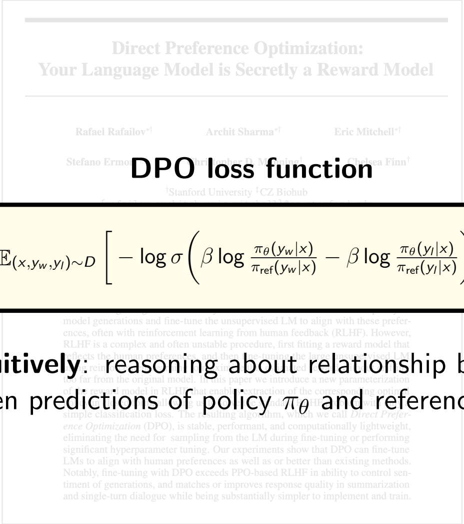
Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

Direct Preference Alignment (DPA) approaches



Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†

Christopher D. Manning† Chelsea Finn†

†Stanford University ‡CZ Biohub

model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that

simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO loss function

$$\mathbb{E}_{(x,y_w,y_l)\sim D}\left[-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

Intuitively: reasoning about relationship between predictions of policy π_θ and reference π_{ref}.

Direct Preference Alignment (DPA) approaches

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†

Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†

†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

These equations are not easy to understand

Direct Preference Alignment (DPA) approaches

DPO loss function

$$\mathbb{E}_{(x,y_w,y_l)\sim D}\left[-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right]$$

Question: What kind of discrete reasoning problems do these losses encode?

The many varieties of DPO

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†
Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†

†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO loss

$$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

The many varieties of DPO

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO loss

$$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

DPO variants

Method	**Objective**
RRHF [91]	$\max\left(0,-\frac{1}{\lvert y_w\rvert}\log\pi_\theta(y_w\lvert x)+\frac{1}{\lvert y_l\rvert}\log\pi_\theta(y_l\lvert x)\right)-\lambda\log\pi_\theta(y_w\lvert x)$
SLiC-HF [96]	$\max(0,\delta-\log\pi_\theta(y_w\lvert x)+\log\pi_\theta(y_l\lvert x))-\lambda\log\pi_\theta(y_w\lvert x)$
DPO [66]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$
IPO [6]	$\left(\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}-\frac{1}{2\tau}\right)^2$
CPO [88]	$-\log\sigma(\beta\log\pi_\theta(y_w\lvert x)-\beta\log\pi_\theta(y_l\lvert x))-\lambda\log\pi_\theta(y_w\lvert x)$
KTO [29]	$-\lambda_w\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-z_{\text{ref}}\right)+\lambda_l\sigma\left(z_{\text{ref}}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$, where $z_{\text{ref}}=\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\beta\mathrm{KL}\left(\pi_\theta(y\lvert x)\lvert\lvert\pi_{\text{ref}}(y\lvert x)\right)\right]$
ORPO [42]	$-\log p_\theta(y_w\lvert x)-\lambda\log\sigma\left(\log\frac{p_\theta(y_w\lvert x)}{1-p_\theta(y_w\lvert x)}-\log\frac{p_\theta(y_l\lvert x)}{1-p_\theta(y_l\lvert x)}\right)$, where $p_\theta(y\lvert x)=\exp\left(\frac{1}{\lvert y\rvert}\log\pi_\theta(y\lvert x)\right)$
R-DPO [64]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}+(\alpha\lvert y_w\rvert-\alpha\lvert y_l\rvert)\right)$
SimPO	$-\log\sigma\left(\frac{\beta}{\lvert y_w\rvert}\log\pi_\theta(y_w\lvert x)-\frac{\beta}{\lvert y_l\rvert}\log\pi_\theta(y_l\lvert x)-\gamma\right)$

from Meng et al. (2024)

- **No reference approaches** (e.g., `CPO`, `ORPO`, *only involves a single model*) versus multi-model, **reference** approaches (`DPO`).

The many varieties of DPO

Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] Archit Sharma[*†] Eric Mitchell[*†]
Stefano Ermon[†‡] Christopher D. Manning[†] Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

DPO loss

$$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

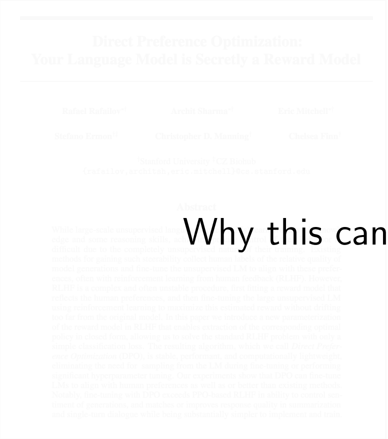
DPO variants

Method	Objective
RRHF [91]	$\max\left(0, -\frac{1}{\lvert y_w\rvert}\log\pi_\theta(y_w\lvert x)+\frac{1}{\lvert y_l\rvert}\log\pi_\theta(y_l\lvert x)\right)-\lambda\log\pi_\theta(y_w\lvert x)$
SLiC-HF [96]	$\max\left(0, \delta-\log\pi_\theta(y_w\lvert x)+\log\pi_\theta(y_l\lvert x)\right)-\lambda\log\pi_\theta(y_w\lvert x)$
DPO [66]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$
IPO [6]	$\left(\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}-\frac{1}{2\tau}\right)^2$
CPO [88]	$-\log\sigma\left(\beta\log\pi_\theta(y_w\lvert x)-\beta\log\pi_\theta(y_l\lvert x)\right)-\lambda\log\pi_\theta(y_w\lvert x)$
KTO [29]	$-\lambda_w\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-z_{\text{ref}}\right)+\lambda_l\sigma\left(z_{\text{ref}}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}\right)$, where $z_{\text{ref}}=\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\beta\text{KL}\left(\pi_\theta(y\lvert x)\Vert\pi_{\text{ref}}(y\lvert x)\right)\right]$
ORPO [42]	$-\log p_\theta(y_w\lvert x)-\lambda\log\sigma\left(\log\frac{p_\theta(y_w\lvert x)}{1-p_\theta(y_w\lvert x)}-\log\frac{p_\theta(y_l\lvert x)}{1-p_\theta(y_l\lvert x)}\right)$, where $p_\theta(y\lvert x)=\exp\left(\frac{1}{\lvert y\rvert}\log\pi_\theta(y\lvert x)\right)$
R-DPO [64]	$-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\lvert x)}{\pi_{\text{ref}}(y_w\lvert x)}-\beta\log\frac{\pi_\theta(y_l\lvert x)}{\pi_{\text{ref}}(y_l\lvert x)}+(\alpha\lvert y_w\rvert-\alpha\lvert y_l\rvert)\right)$
SimPO	$-\log\sigma\left(\frac{\beta}{\lvert y_w\rvert}\log\pi_\theta(y_w\lvert x)-\frac{\beta}{\lvert y_l\rvert}\log\pi_\theta(y_l\lvert x)-\gamma\right)$

from Meng et al. (2024)

Questions: How are all these variations related to one another, nature of the space of losses?

The many varieties of DPO


Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

DPO loss

$$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

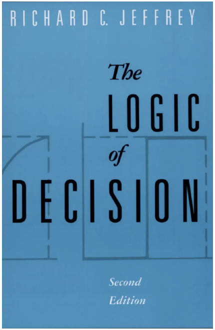
DPO variants

from Meng et al. (2024)

Why this can be frustrating

Haven't these semantic questions been looked at before?

Analytic philosophy: Much work on the semantics of pairwise preference, rich languages for expressing ideas.



(Jeffrey, 1965)

THE STATUS OF VARIOUS PREFERENCE PRINCIPLES

Preference Principle	Von Wright	Chisholm Sosa	Martin	$P^{\sharp}$	$P^{\star}$	P^{w}
1. $pPq \to \sim(qPp)$	✓	✓	✓	+	+	+
2. $(pPq \;\&\; qPr) \to pPr$	✓	✓	✓	+	+	+
3. $pPq \to \sim qP\sim p$		x	✓	(+)[1]	+	+
4. $\sim qP\sim p \to pPq$		x	✓	(+)[1]	+	+
5. $pPq \to (p \;\&\; \sim q)\; P(\sim p \;\&\; q)$	✓	x		+	+	+
6. $(p \;\&\; \sim q)\; P(\sim p \;\&\; q) \to pPq$	✓	x		+	+	+
7. $[\sim(pP\sim p) \;\&\; \sim(\sim pPp) \;\&\; \sim(qP\sim q) \;\&\; \sim(\sim qPq)] \to [\sim(pPq) \;\&\; \sim(qPp)]$	✓	✓		+	+	+
8. $[\sim(qP\sim q) \;\&\; \sim(\sim qPq) \;\&\; pPq] \to pP\sim p$		✓		+	+	—
9. $[\sim(qP\sim q) \;\&\; \sim(\sim qPq) \;\&\; qP\sim p] \to pP\sim p$		✓		+	+	—
10. $pPq \to [(p \;\&\; r)\; P(q \;\&\; r) \;\&\; (p \;\&\; \sim r)\; P(q \;\&\; \sim r)]$	✓			—	—	+
11. $[(p \;\&\; r)\; P(q \;\&\; r) \;\&\; (p \;\&\; \sim r)\; P(q \;\&\; \sim r)] \to pPq$	✓			(+)[2]	(+)[3]	+
12. $[\sim(pPq) \;\&\; \sim(qPr)] \to \sim(pPr)$		✓		+	+	—
13. $(pPr \vee qPr) \to (p \vee q)\; Pr$			✓	—	—	—
14. $(p \vee q)\; Pr \to [pPr \;\&\; qPr]$	✓			—	—	—
15. $[pPr \;\&\; qPr] \to (p \vee q)\; Pr$	✓			—	—	—
16. $(p \vee q)\; Pr \to (pPr \vee qPr)$				—	—	—
17. $pP(q \vee r) \to (pPq \;\&\; pPr)$			✓	—	—	—
18. $(pPq \;\&\; pPr) \to pP(q \vee r)$				—	—	—
19. $(pPr \;\&\; qPr) \to (p \;\&\; q)\; Pr$				—	—	—

Semantic foundations for the logic of preference Rescher (1967)

The language of machine learning

Loss functions

$$\boxed{-\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)}$$

The language of machine learning

Loss functions

$$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

- Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

The language of machine learning

Loss functions

$$\boxed{-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)}$$

Specification or theory of preference?

- Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

The language of machine learning

Loss functions

$$\boxed{-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)}$$

$\uparrow$

Specification or theory of preference?

- Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

Broader goal: High-level modeling languages for specifying and better understanding LLMs and their algorithms.

The language of machine learning

Loss functions

$$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

$\uparrow$

Specification or theory of preference?

Formalization of preference losses

- Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

Broader goal: High-level modeling languages for specifying and better understanding LLMs and their algorithms.

The language of machine learning

Loss functions

$$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

$$\uparrow$$

Specification or theory of preference?

Going away from these opaque equations

- Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

Broader goal: High-level modeling languages for specifying and better understanding LLMs and their algorithms.

Preference learning as a discrete reasoning problem

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Preference learning as a discrete reasoning problem

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Two models, four predictions

Preference learning as a discrete reasoning problem

Loss Function

$$\boxed{-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)}$$

Two models, four predictions

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

High-level model behavior

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Decompilation ⟷ Compilation

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Loss Function

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Decomposition $\longleftrightarrow$ Compilation

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Decompilation $\longleftrightarrow$ Compilation

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

  1. **Compilation**: Translating specifications into loss, well studied.

Preference learning as a discrete reasoning problem

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Loss Function

$$-\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Decompilation $\longleftrightarrow$ Compilation

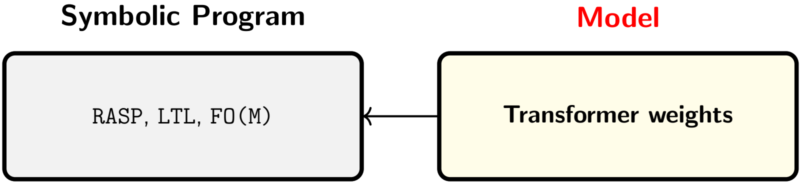
- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?
  1. **Compilation**: Translating specifications into loss, well studied.
  2. **Decompilation**:Losses to specifications (inverse), less explored.

Formal analysis via decompilation in general

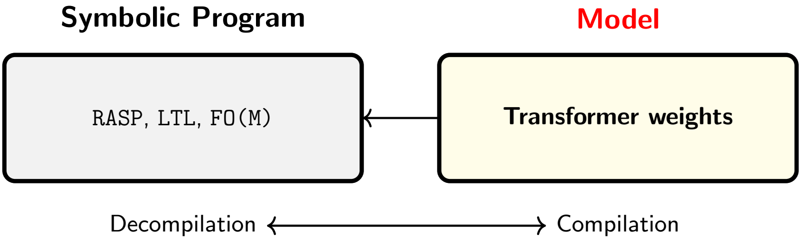
Model

Transformer weights

Formal analysis via decompilation in general



Formal analysis via decompilation in general



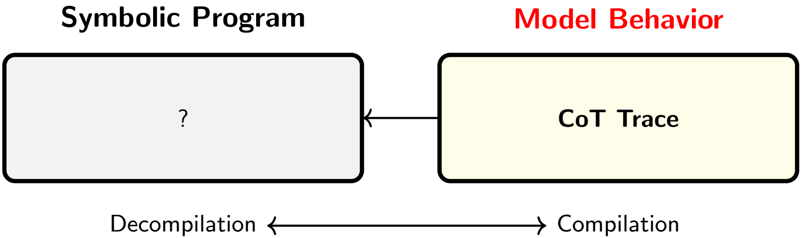
- We know what the *target languages* are (Weiss et al., 2021; Merrill and Sabharwal, 2023; Yang and Chiang, 2024), how to compile, decompile (Friedman et al., 2023).

Formal analysis via decompilation in general

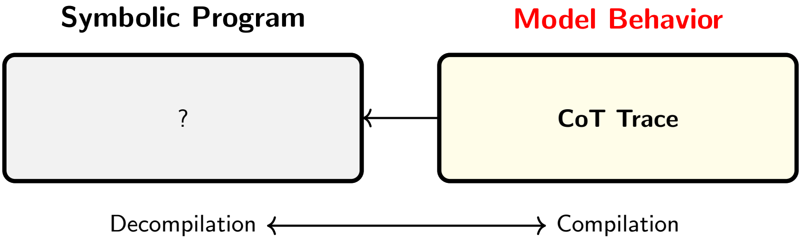
Model Behavior

CoT Trace

Formal analysis via decompilation in general



Formal analysis via decompilation in general



- Not always clear what the target language is or should be.

Language model programming: **ESSLLI 2025**

Lecturers

Kyle Richardson (Allen Institute for AI)

Gijs Wijnholds (Leiden Institute of Advanced Computer Science)

Slides

lecture 1: course overview, **language modeling basics**, transformers, RASP.

lecture 2: declarative approaches to **model training and fine-tuning**, the semantic loss and weighted model counting, other approaches.

lecture 3: high-level programming techniques for direct preference alignment and LLM alignment, formal characterizations of *known* loss functions.

lecture 4: declarative and probabilistic approaches to test-time inference, LLM self-correction, consistency, distilling LLMs to tractable models, logic programming.

lecture 5: Advanced prompting, chain-of-thought, imperative model programming, (discrete) probabilistic programming.

background logic notes, extended notes on transformers

extra lectures Prompting as programming, Grammar-constrained decoding

Helpful Resources

Below are some pointers to code resources:

- **languages** [scallop], [problog], [pyDatalog], [lmql],[rasp], [NumPy Rasp], [deepproblog]
- **automated reasoning tools/circuits** [Z3 solver], [python-sat], [pysdd], [cirkit]
- **NLP and general ML** [transformers], [PyTorch], [pylon-lib], [hf datasets], [hf hub]
- **other useful utilities** [sympy]

Useful tutorials: Transformers from scratch (*some examples/ideas used in lecture 1*), Lectures on Probabilistic Programming, Tractable Probabilistic Models

https://github.com/yakazimir/LMProgramming

What is the right programming language for preference?

Declarative models of preference

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

Declarative models of preference

Model predicts loser

Model predicts winner

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

Conceptually: Model predications are logical propositions, Boolean variables inside of formulas, weighted by prediction probability.

Declarative models of preference

Model predicts loser

Model predicts winner

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$w(\mathtt{M}(\mathtt{x},\mathtt{y})) = \pi_{\mathtt{M}}(y \mid x)$$

Conceptually: Model predications are logical propositions, Boolean variables inside of formulas, weighted by prediction probability.

Declarative models of preference

Model predicts loser

Model predicts winner

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Conceptually: Predictions are connected through Boolean operators, express constraints on predictions; ρ_θ as formulas.

Uncovering the natural logic of these algorithms

Model predicts loser

Model predicts winner

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

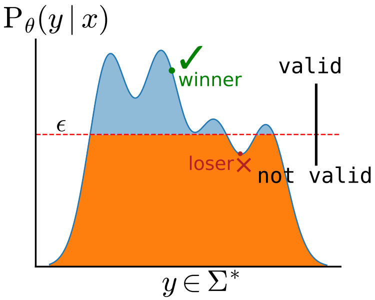
Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

Uncovering the natural logic of these algorithms

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

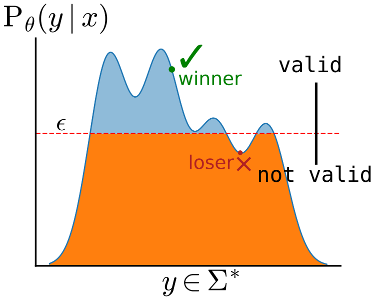


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

Uncovering the natural logic of these algorithms

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

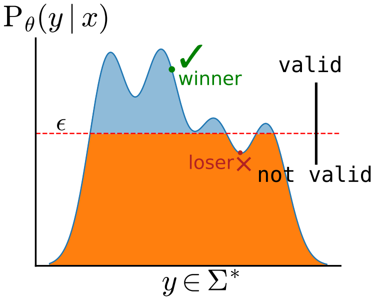


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

Uncovering the natural logic of these algorithms

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```



Observation: The second program is more strict than the first, involves semantic entailment.

Compilation and decompilation again

Compilation

P

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$\ell(D,\theta) = -\log \underbrace{P_\theta(\mathsf{P} \mid D,\theta)}_{\text{probabilistic logic}}$$

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Compilation and decompilation again

Compilation

P

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$\ell(D,\theta) = -\log \underbrace{P_\theta(\mathsf{P} \mid D,\theta)}_{\text{probabilistic logic}}$$

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

What we did: defined a novel probabilistic logic for preference modeling,

note: logic useful not only for learning and loss.

Compilation and decompilation again

Decompilation

P

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$\ell_{\texttt{CPO}} = -\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}\right)$$

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Compilation and decompilation again

Decompilation

P

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$\ell_{\texttt{CPO}} = -\log \sigma\left(\log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \right)$$

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\underbrace{\ell_{\texttt{CPO}}(D,\theta) = -\log P_\theta(\mathsf{P} \mid D,\theta)}_{\text{correctness property}}$$

Compilation and decompilation again

Decompilation

P

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$$\ell_{\texttt{CPO}} = -\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}\right)$$

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\underbrace{\ell_{\texttt{CPO}}(D,\theta) = -\log P_\theta(\mathsf{P} \mid D,\theta)}_{\text{correctness property}}$$

The second thing we did: Defined a mechanical procedure for decompilation, proved its correctness, invariance to choice of f.

Illustration of approach and results

Input Loss $\ell_{\texttt{ORPO}}$

$$-\log \sigma\left(\log \frac{\text{Odds}_\theta(y_w|x)}{\text{Odds}_\theta(y_l|x)} \right)$$

Illustration of approach and results

Input Loss $\ell_{\texttt{ORPO}}$

$$-\log\sigma\left(\log\frac{\mathrm{Odds}_\theta(y_w|x)}{\mathrm{Odds}_\theta(y_l|x)}\right)$$

$$\frac{\rho_\theta^t}{\rho_\theta^b} = \frac{P_\theta(y_w|x)(1-P_\theta(y_l|x))}{P_\theta(y_l|x)(1-P_\theta(y_w|x))}$$

Core loss equation

Illustration of approach and results

Input Loss $\ell_{\texttt{ORPO}}$

$$-\log\sigma\left(\log\frac{\mathsf{Odds}_\theta(y_w|x)}{\mathsf{Odds}_\theta(y_l|x)}\right)$$

$$\frac{\rho_\theta^t}{\rho_\theta^b} = \frac{P_\theta(y_w|x)(1-P_\theta(y_l|x))}{P_\theta(y_l|x)(1-P_\theta(y_w|x))}$$

Core loss equation

$$\textsc{Sem}(\rho_\theta^t) = \texttt{M}(\mathsf{x},\mathsf{y}_w) \land \neg\texttt{M}(\mathsf{x},\mathsf{y}_l)$$
$$\textsc{Sem}(\rho_\theta^b) = \texttt{M}(\mathsf{x},\mathsf{y}_l) \land \neg\texttt{M}(\mathsf{x},\mathsf{y}_w)$$

Compositional translation

Illustration of approach and results

Input Loss $\ell_{\texttt{ORPO}}$

$$-\log \sigma\left(\log \frac{\mathrm{Odds}_\theta(y_w|x)}{\mathrm{Odds}_\theta(y_l|x)}\right)$$

Core loss equation

$$\frac{\rho_\theta^t}{\rho_\theta^b} = \frac{P_\theta(y_w|x)(1-P_\theta(y_l|x))}{P_\theta(y_l|x)(1-P_\theta(y_w|x))}$$

Compositional translation

$$\textsc{Sem}(\rho_\theta^t) = \texttt{M}(\mathsf{x}, \mathsf{y}_w) \land \neg\texttt{M}(\mathsf{x}, \mathsf{y}_l)$$
$$\textsc{Sem}(\rho_\theta^b) = \texttt{M}(\mathsf{x}, \mathsf{y}_l) \land \neg\texttt{M}(\mathsf{x}, \mathsf{y}_w)$$

Preference structure $\overline{\mathsf{P}}$

$\mathsf{P} := \texttt{Implies}($
$\quad \texttt{M}(\mathsf{x},\mathsf{y}_l),\ \texttt{M}(\mathsf{x},\mathsf{y}_w))$
$\mathsf{P}_{\texttt{C}} := \texttt{XOR}(\texttt{M}(\mathsf{x},\mathsf{y}_l),\texttt{M}(\mathsf{y}_w)))$
$\mathsf{P}_{\texttt{A}} := \bot$

Illustration of approach and results

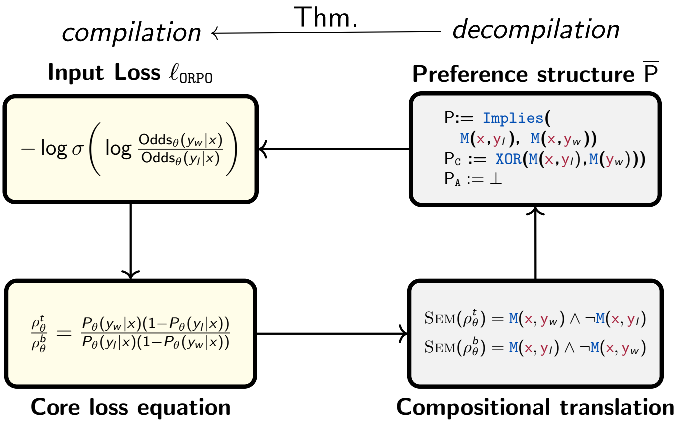
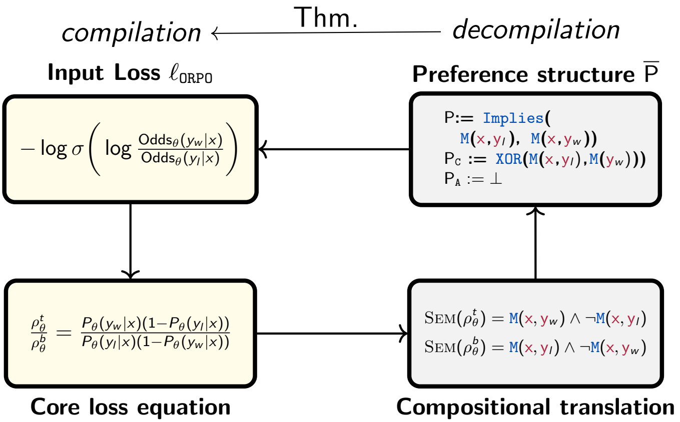


Illustration of approach and results



- **Preference structure**, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

Illustration of approach and results

compilation $\xleftarrow{\text{Thm.}}$ *decompilation*

Input Loss $\ell_{\texttt{ORPO}}$

$$-\log\sigma\left(\log\frac{\text{Odds}_\theta(y_w|x)}{\text{Odds}_\theta(y_l|x)}\right)$$

Preference structure $\overline{\mathsf{P}}$

$\mathsf{P} := \texttt{Implies(}$
$\quad \texttt{M}(x, y_l)\texttt{,}\ \texttt{M}(x, y_w)\texttt{)}$
$\mathsf{P}_C := \texttt{XOR}(\texttt{M}(x, y_l), \texttt{M}(y_w)))$
$\mathsf{P}_A := \bot$

question: Why is this useful to do?

$$\frac{\rho^t_\theta}{\rho^b_\theta} = \frac{P_\theta(y_w|x)(1-P_\theta(y_l|x))}{P_\theta(y_l|x)(1-P_\theta(y_w|x))}$$

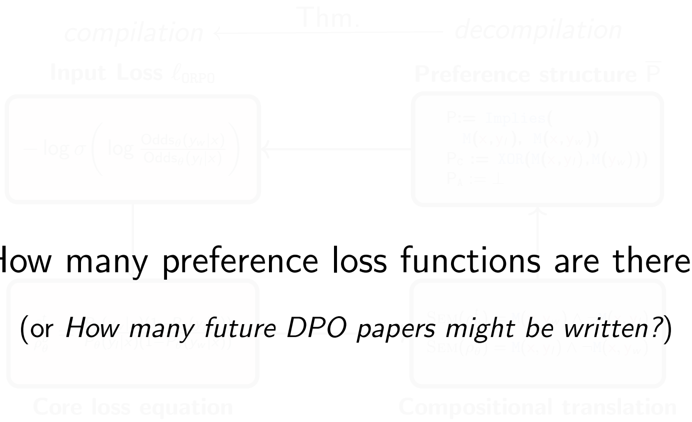
Core loss equation

$\textsc{Sem}(\rho^t_\theta) = \texttt{M}(x, y_w) \land \neg\texttt{M}(x, y_l)$
$\textsc{Sem}(\rho^b_\theta) = \texttt{M}(x, y_l) \land \neg\texttt{M}(x, y_w)$

Compositional translation

- **Preference structure**, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

Illustration of approach and results



How many preference loss functions are there?

(or *How many future DPO papers might be written?*)

- **Preference structure**, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

Why is this useful? understanding the space

$P^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$P^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

Boolean functions, **2 variables**

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Why is this useful? understanding the space

$P^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$P^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

Boolean functions, **2 variables**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

- Every program (in our logic) is pair of Boolean functions (in n variables), corr. to ✓ and ✗, leads to 4^{2^n} possible loss functions.

Why is this useful? understanding the space

$P^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$P^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

Boolean functions, **2 variables**

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Loss creation will end up being equivalent to drawing different sets of ✓ s and ✗ (or blank marks) in a truth table.

Why is this useful? understanding the space

$\mathsf{P}^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$\mathsf{P}^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

Boolean functions, **2 variables**

M(x, y_w)	M(x, y_l)	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

no reference: 256 losses

Loss creation will end up being equivalent to drawing different sets of ✓ s and ✗ (or blank marks) in a truth table.

Loss functions as truth tables

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

4 variables

Ref(x, y_w)	M(x, y_l)	Ref(x, y_l)	M(x, y_w)
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

w/ reference: 4,294,967,296 losses

Loss functions as truth tables

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

4 variables

Ref(x, y_w)	M(x, y_l)	Ref(x, y_l)	M(x, y_w)
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

w/ reference: 4,294,967,296 losses

answer: loads.

Loss functions as truth tables

$\texttt{Ref}(x, y_w)$	$\texttt{M}(x, y_l)$	$\texttt{Ref}(x, y_l)$	$\texttt{M}(x, y_w)$
F	F	F	F
F	F	F	T
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
```

4 variables

w/ reference: 4,294,967,296 losses

question: How are losses related to one another?

Why is this useful? understanding the structure

$\mathsf{P}^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$\mathsf{P}^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

semantics: $\mathsf{P}^{(2)} \models \mathsf{P}^{(1)}$

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Proposition (Xu et al., 2018): Loss behavior is monotonic w.r.t semantic entailment: if $\mathsf{P}^{(2)} \models \mathsf{P}^{(1)}$ then $\ell(D, \theta, \mathsf{P}^{(2)}) \geq \ell(D, \theta, \mathsf{P}^{(1)})$.

Why is this useful? understanding the structure

$P^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$P^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

semantics: $P^{(2)} \models P^{(1)}$

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Proposition (Xu et al., 2018): Loss is equivalent under semantic equivalence: If $P^{(2)} \equiv P^{(1)}$ then $\ell(D, \theta, P^{(2)}) = \ell(D, \theta, P^{(1)})$.

Why is this useful? understanding the structure

$P^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$P^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

semantics: $P^{(2)} \models P^{(1)}$

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Theorem: $\ell(D, \theta, P^{(2)}) > \ell(D, \theta, P^{(1)})$ (the loss of $P^{(1)}$ is contained in the loss of $P^{(2)}$).

Why is this useful? understanding the structure

$\mathrm{P}^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

semantics: $\mathrm{P}^{(2)} \models \mathrm{P}^{(1)}$

$\mathrm{P}^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathrm{P}^{(1)}$	$\mathrm{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

answer: Losses are related through their semantics

Theorem: $\ell(D, \theta, \mathrm{P}^{(2)}) \geq \ell(D, \theta, \mathrm{P}^{(1)})$ (the loss of $\mathrm{P}^{(1)}$ is contained in the loss of $\mathrm{P}^{(2)}$).

Why is this useful? understanding the structure

$\mathsf{P}^{(1)}$

```
Implies(
  M(x,y_l),M(x,y_w)
)
```

$\mathsf{P}^{(2)}$

```
And(
  M(x,y_w),
  Not(M(x,y_l)))
```

semantics: $\mathsf{P}^{(2)} \models \mathsf{P}^{(1)}$

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

Practical strategy: Start with empirically successful losses, modify semantics (make more or less constrained), then experiment accordingly.

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

DPO Loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

DPO Loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

modify

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  M(x,y_w)
)
```

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

DPO Loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

modify

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  M(x,y_w)
)
```

Novel loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)\pi_{\text{ref}}(y_l|x)(1-\pi_\theta(y_l|x))}{\pi_\theta(y_l|x)\pi_{\text{ref}}(y_w|x)(1-\pi_\theta(y_w|x))}\right)$$

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

DPO Loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

modify

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  M(x,y_w)
)
```

Novel loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)\pi_{\text{ref}}(y_l|x)(1-\pi_\theta(y_l|x))}{\pi_\theta(y_l|x)\pi_{\text{ref}}(y_w|x)(1-\pi_\theta(y_w|x))}\right)$$

- High-level programming language for defining new losses.

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

DPO Loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)}-\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

questions: How does our logic work? What do we see?

Modify

```
Implies(
  And(M(x,y_l),Ref(x,y_w)),
  M(x,y_w)
)
```

Novel loss

$$-\log\sigma\left(\log\frac{\pi_\theta(y_w|x)\pi_{\text{ref}}(y_l|x)(1-\pi_\theta(y_l|x))}{\pi_\theta(y_l|x)\pi_{\text{ref}}(y_w|x)(1-\pi_\theta(y_w|x))}\right)$$

How does the logic work? **compilation**

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

How does the logic work? **compilation**

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \texttt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg\texttt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \mathtt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

- Formula probability computed as a weighted count $\sum \checkmark_w$ (Chavira and Darwiche, 2008), loss is $-\log$, *semantic loss* (Xu et al., 2018).

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \mathtt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := \underbrace{-\log \sigma\left(\log \frac{\sum \checkmark}{\sum \times}\right)}_{\textit{arbitrary}\ \times_w}$$

How does the logic work? **compilation**

M(x, y_w)	M(x, y_l)	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{ \checkmark, \times \}_w := \prod_{w \models \texttt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \texttt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left(\log \frac{\sum \checkmark_w}{\sum \times_w} \right)$$

$$= \underbrace{-\log \sigma \left(\log \frac{\pi_\theta(y_w \mid x)(1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))} \right)}_{\ell_{\texttt{ORPO}},\ P_\theta(\mathsf{P} \mid \textbf{one hot})}$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,yl), M(x,yw)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \mathtt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left(\log \frac{\sum \checkmark_w}{\sum \times_w} \right)$$

$$= \underbrace{-\log \sigma \left(\log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} \right)}_{\ell_{\mathtt{CPO}},\ \sim P_\theta(\mathrm{P} \mid \textbf{one true})}$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

observation: losses differ in hard constraints

$$\underbrace{\ell_x}_{\text{column}} := -\log\sigma\left(\log\frac{\sum ✓_w}{\sum X_w}\right)$$

$$= \underbrace{-\log\sigma\left(\log\frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)}\right)}_{\ell_{\mathtt{CPO}},\ \sim P_\theta(\mathrm{P}|\textbf{one true})}$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a <u>valid</u> generation, it should deem the winner to be <u>valid</u> too.

$$\{\checkmark, \times\}_w := \prod_{w \models \mathtt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

Loss	**Representation** $\overline{\mathsf{P}}$
CE	$\mathsf{P} := \mathbf{M}(x,y_w),\ \mathsf{P_C} := \bot$
CEUnl	$\mathsf{P} := \mathbf{And}(\mathbf{M}(x,y_w),\ \mathbf{Not}(\mathbf{M}(x,y_l)))$ $\mathsf{P_C} := \bot$
CPO	;; core semantic formula $\mathsf{P} := \mathbf{Implies}(\mathbf{M}(x,y_l), \mathbf{M}(x,y_w))$;; one-true constraint $\mathsf{P_C} := \mathbf{Or}(\mathbf{M}(x,y_l),\ \mathbf{M}(x,y_w))$
ORPO	$\mathsf{P} := \mathbf{Implies}(\mathbf{M}(x,y_l),\mathbf{M}(x,y_w))$;; one-hot constraint $\mathsf{P_C} := \mathbf{XOR}(\mathbf{M}(x,y_l),\ \mathbf{M}(x,y_w))$

How does the logic work? **compilation**

$\texttt{M}(\mathrm{x}, \mathrm{y}_w)$	$\texttt{M}(\mathrm{x}, \mathrm{y}_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{ \checkmark, \times \}_w := \prod_{w \models \texttt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \texttt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

- **Preference structure**: equivalent way of expressing truth table representations (Richardson et al., 2025),

$$\overline{\mathrm{P}} := \Big(\underbrace{\mathrm{P}}_{\text{core}}, \underbrace{\mathrm{P}_{\mathbf{C}}, \mathrm{P}_{\mathbf{A}}}_{\text{constraints}} \Big)$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \mathtt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma\left(\log \frac{\sum \checkmark_w}{\sum \times_w} \right)$$

$$= \underbrace{-\log \sigma\left(\log \frac{\pi_\theta(y_l \mid x)\pi_\theta(y_w \mid x) + (1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))} \right)}_{\textbf{novel loss without constraints, } P_\theta(\mathsf{P} \mid \top)}$$

How does the logic work? **compilation**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

note: $\mathtt{M}(x, y_l) \rightarrow \mathtt{M}(x, y_w) \equiv \neg\mathtt{M}(x, y_l) \vee \mathtt{M}(x, y_w)$

$$\underbrace{\ell_x}_{\text{column}} := -\log\sigma\left(\log\frac{\sum ✓_w}{\sum ✗_w}\right)$$

$$= \underbrace{-\log\sigma\left(\log\frac{\pi_\theta(y_l \mid x)\pi_\theta(y_w \mid x) + (1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))}\right)}_{\textbf{novel loss without constraints, } P_\theta(\mathrm{P} \mid \top)}$$

What properties do real losses have?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \times\}_w := \prod_{w \models \texttt{M}(x,y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg \texttt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma\left(\log \frac{\sum \checkmark_w}{\sum \times_w}\right)$$

$$= \underbrace{-\log \sigma\left(\log \frac{\pi_\theta(y_l \mid x)\pi_\theta(y_w \mid x) + (1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))}\right)}_{\textbf{novel loss without constraints, } P_\theta(\mathsf{P} \mid \top)}$$

Mapping out these loss spaces semantically

What properties do real losses have?

M(x, y_w)	M(x, y_l)	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

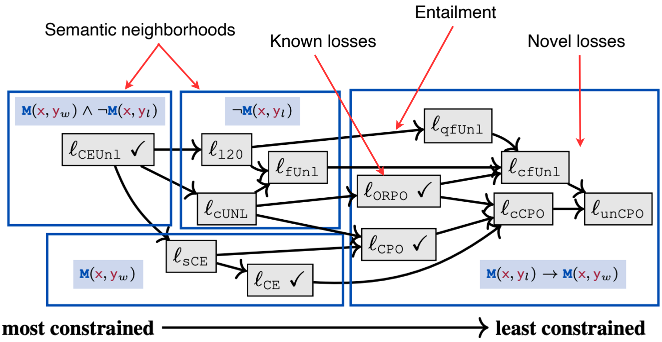
```
Implies(
  M(x,y_l), M(x,y_w)
)
```

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\underbrace{\ell_x}_{\text{column}} := -\log\sigma\left(\log\frac{\sum ✓_w}{\sum ✗_w}\right)$$

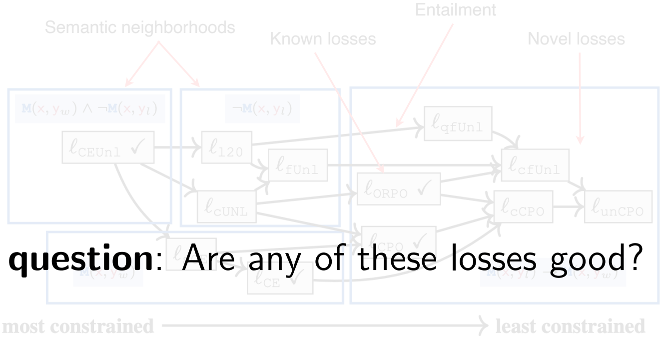
$$= \underbrace{-\log\sigma\left(\log\frac{\pi_\theta(y_l\mid x)\pi_\theta(y_w\mid x)+(1-\pi_\theta(y_l\mid x))}{\pi_\theta(y_l\mid x)(1-\pi_\theta(y_w\mid x))}\right)}_{\textbf{novel loss without constraints, } P_\theta(\mathsf{P}\mid\top)}$$

The no reference loss landscape



- **Loss lattice**: semantic structure of space, ordering.

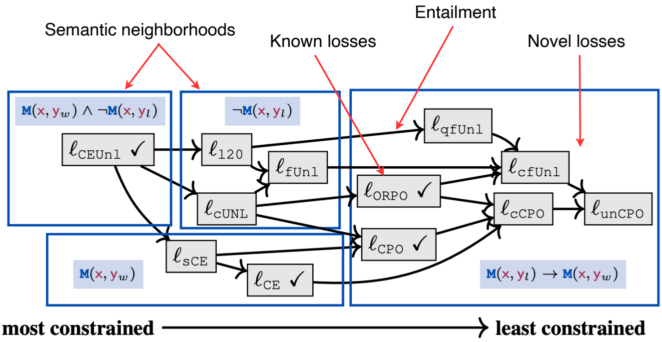
The no reference loss landscape

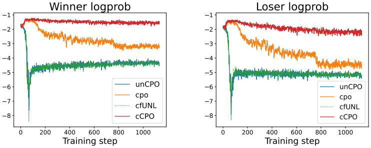


question: Are any of these losses good?

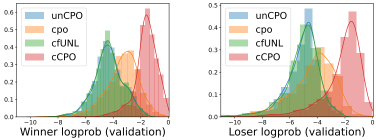
Training dynamics **Inference**

The no reference loss landscape



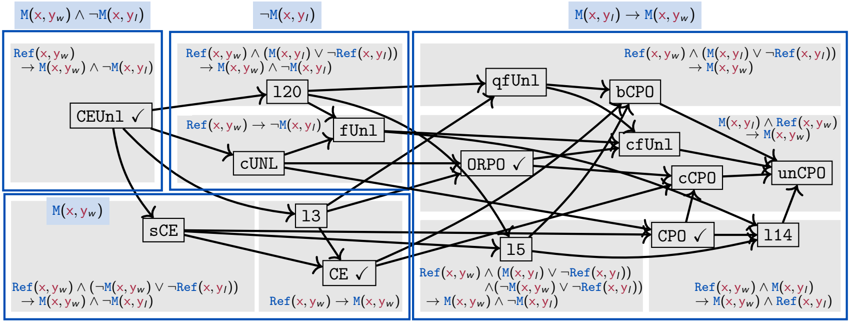


Training dynamics



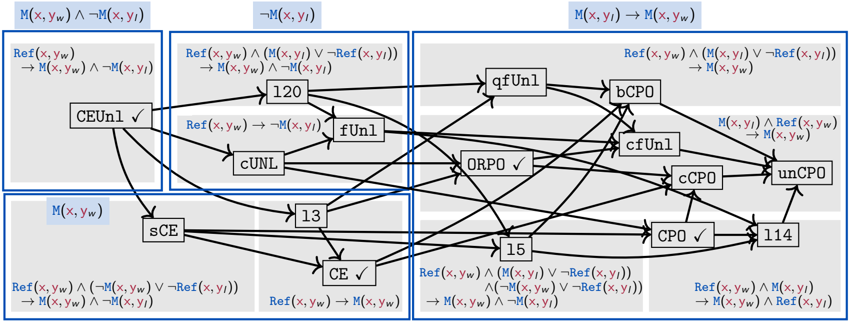
Inference

The full landscape, **reference approaches**



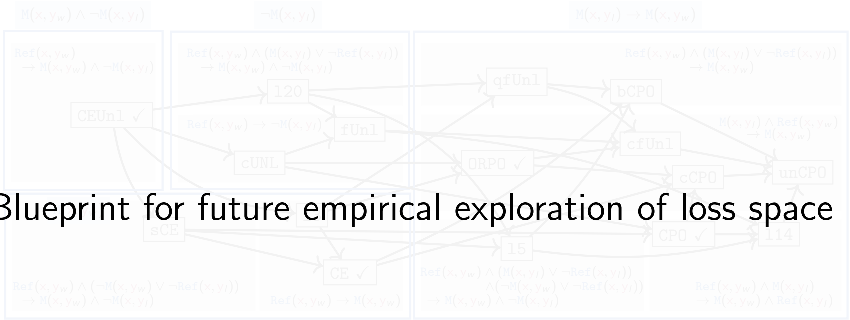
- The semantics of DPO-style reference losses can be straightforwardly computed from no reference approaches.

The full landscape, **reference approaches**



- Many new losses to explore and experiment with!

The full landscape, **reference approaches**



Blueprint for future empirical exploration of loss space

Conclusions

- New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.

Conclusions

- New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.

  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

Conclusions

- New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.

  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

The procedure: write a (high-level) symbolic program, or modify an existing one, compile into a loss and experiment (then repeat)

Conclusions

- New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.

  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

  The procedure: write a (high-level) symbolic program, or modify an existing one, compile into a loss and experiment (then repeat)

- **many other areas to look at**: *analysis of transformers*, *semantics of data*, *reinforcement learning*, *chain-of-thought*, *LLM agents* ...

Thank you.

Adding a reference model

```
P:= Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Whenever the model being tuned deems the loser to be a valid generation and the reference model deems the winner to be valid, the tuned model should deem the winner to be valid too, and the reference should deem the loser to be valid.

Adding a reference model

```
P:= Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

Whenever the model being tuned deems the loser to be a valid generation and the reference model deems the winner to be valid, the tuned model should deem the winner to be valid too, and the reference should deem the loser to be valid.

- **Peculiar semantics**, but the logic makes sense, e.g., we want to maximize

$$\sigma\left(\log\frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} - \log\frac{\pi_{\text{ref}}(y_w \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)$$

negating left side of implication (i.e., making $\texttt{M}(x, y_l)$ and $\texttt{Ref}(x, y_w)$ false) and making the right side true is logical.

References I

Bogin, B., Yang, K., Gupta, S., Richardson, K., Bransom, E., Clark, P., Sabharwal, A., and Khot, T. (2024). Super: Evaluating agents on setting up and executing tasks from research repositories. *Proceedings of EMNLP*.

Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799.

Chen, J., Yuan, S., Ye, R., Majumder, B. P., and Richardson, K. (2023). Put your money where your mouth is: Evaluating strategic planning and execution of llm agents in an auction arena. *arXiv preprint arXiv:2310.05746*.

Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., and Yang, Y. (2024). Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*.

Friedman, D., Wettig, A., and Chen, D. (2023). Learning transformer programs. *Advances in Neural Information Processing Systems*, 36:49044–49067.

Jeffrey, R. C. (1965). *The logic of decision*. University of Chicago press.

Ji, J., Liu, M., Dai, J., Pan, X., Zhang, C., Bian, C., Chen, B., Sun, R., Wang, Y., and Yang, Y. (2024). Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.

Meng, Y., Xia, M., and Chen, D. (2024). Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Merrill, W. and Sabharwal, A. (2023). A logic for expressing log-precision transformers. *Advances in neural information processing systems*, 36:52453–52463.

References II

Rescher, N. (1967). *The logic of decision and action*. University of Pittsburgh Pre.

Richardson, K., Srikumar, V., and Sabharwal, A. (2025). Understanding the logic of direct preference alignment through logic. *Proceedings of ICML*.

Weiss, G., Goldberg, Y., and Yahav, E. (2021). Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Broeck, G. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *International Conference on Machine Learning*, pages 5498–5507.

Yang, A. and Chiang, D. (2024). Counting like transformers: Compiling temporal counting logic into softmax transformers. *arXiv preprint arXiv:2404.04393*.

Yang, R., Chen, J., Zhang, Y., Yuan, S., Chen, A., Richardson, K., Xiao, Y., and Yang, D. (2025). Selfgoal: Your language agents already know how to achieve high-level goals. *Proceedings of NAACL*.

Zhang, Y., Yuan, S., Hu, C., Richardson, K., Xiao, Y., and Chen, J. (2024). Timearena: Shaping efficient multitasking language agents in a time-aware simulation. *Proceedings of ACL*.