

Understanding the Logic of Generative AI through Logic

Kyle Richardson

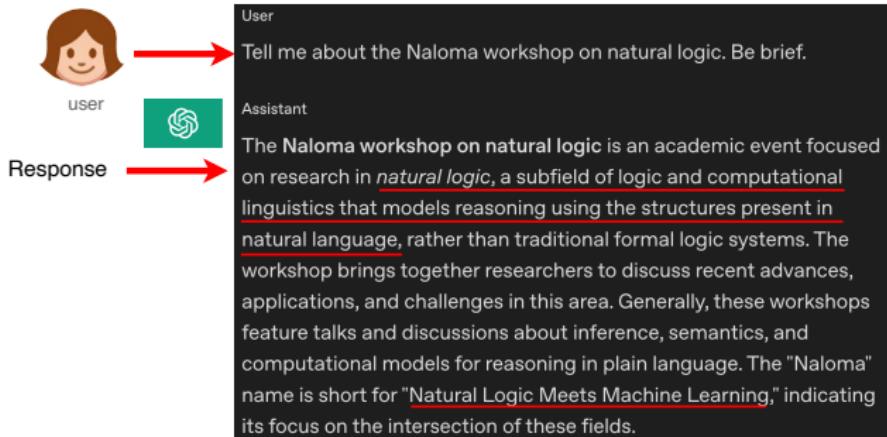
Allen Institute for AI (AI2)

August 2025

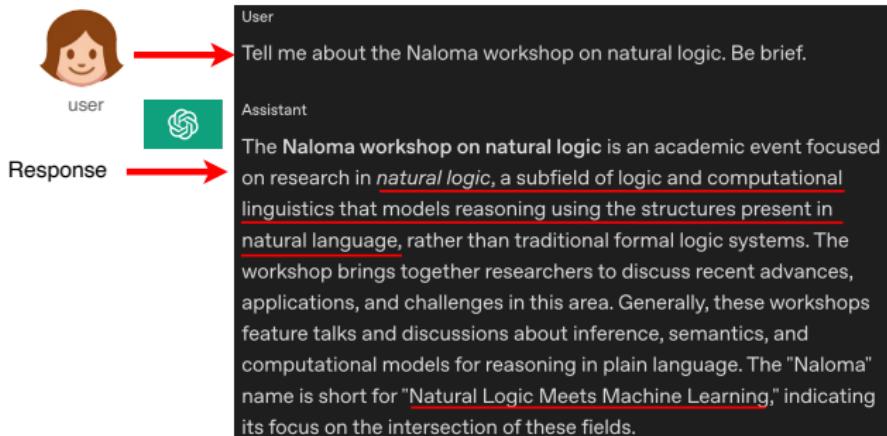
Collaborators: Ashish Sabharwal (AI2), Vivek Srikumar (University of Utah)



General purpose large language models (LLMs)

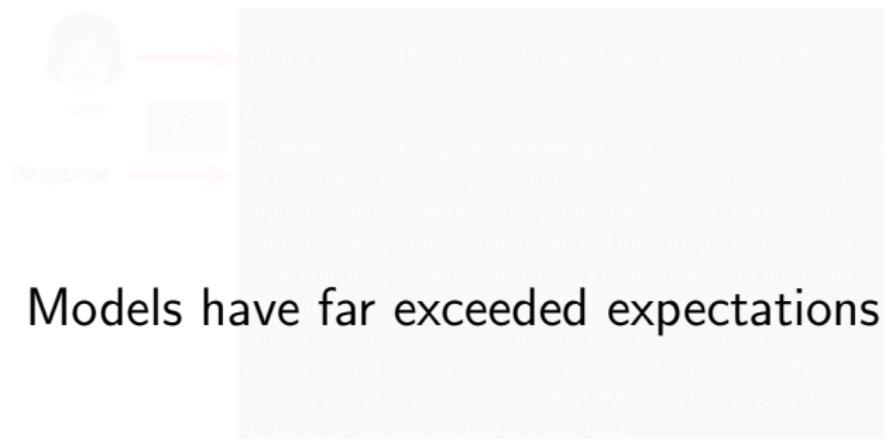


General purpose large language models (LLMs)



- ▶ **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

Computational models have exceeded expectations. The “Natoma Implementation for Natural Logic Meets Machine Learning” indicates the range of the interaction in these terms:

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

Language models as agent simulators

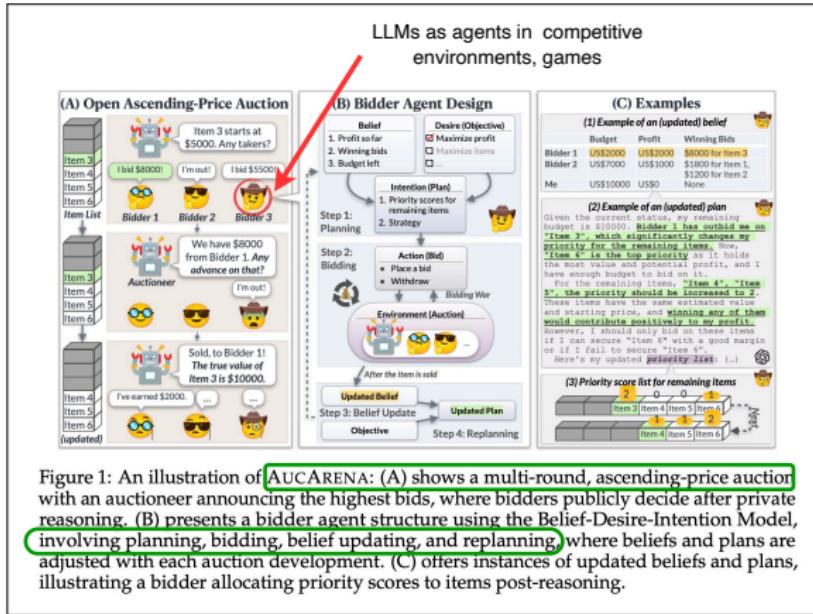
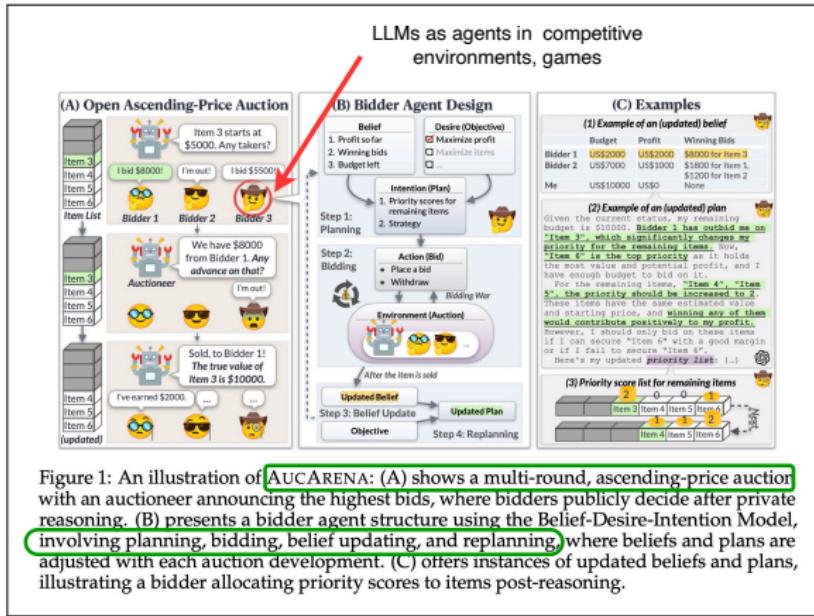


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief 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



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

Language models as part of complex systems

The screenshot illustrates a workflow for implementing an encoder-only model for multiple-choice QA. It starts with a query in the 'ML Experiment Execution Engine' interface, which then triggers a series of steps:

- Machine learning experiment:** A user icon and a query: "Implement an encoder-only model using transformers that can do multiple-choice QA".
- Step 1:** A system thought: "To implement an encoder-only model for multiple-choice question answering (QA) using the transformers library from Hugging Face, I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to handle multiple-choice QA tasks. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the `transformers` library for this purpose." An arrow points from this step to the ChatGPT section.
- ChatGPT:** A response: "Action [execute]".

```
from transformers import AutoModelForMultipleChoice, AutoTokenizer

model_name = 'bert-base-uncased'

# Load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMultipleChoice.from_pretrained(model_name)

print("Model and tokenizer loaded successfully.")
```
- Model generated code:** An observation: "Model and tokenizer loaded successfully." An arrow points from this step to the Step 2 section.
- Step 2:** A system thought: "Define an example implementation to start:
Model Name : bert-base-uncased
Tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMultipleChoice.from_pretrained(model_name)
print('Model and tokenizer loaded successfully')
Model, Tokenizer and tokenizer loaded successfully"
An arrow points from this step to the Jupyter notebook section.
- Jupyter notebook:** A screenshot of a Jupyter notebook titled "automated_ml_notebook>Last Checkpoint: 23 minutes ago". It shows the same code and output as the previous step, along with a progress bar and some statistics:

```
Model, Tokenizer and tokenizer loaded successfully
```

Model	40/440 [00:00<0:00, 0.82MB/s]
Tokenizer	100% [██████████] 5150/5150 [00:00<0:00, 69.0KB/s]
ocab_size	100% [██████████] 230023/230023 [00:00<0:00, 119KB/s]
tokenization	100% [██████████] 400000/400000 [00:00<0:00, 100KB/s]
modelInitialization	100% [██████████] 400000/400000 [00:00<0:00, 98.0KB/s]

Some weights of BertForMultipleChoice were not initialized from the model checkpoint at bert-base-uncased as they are newly initialized. (classification_head, 'classifier.weight'). You should probably TRAIN this model on a downstream task to be able to use it for predictions and inference.
Model and tokenizer loaded successfully

Model and tokenizer loaded successfully

Model and tokenizer loaded successfully

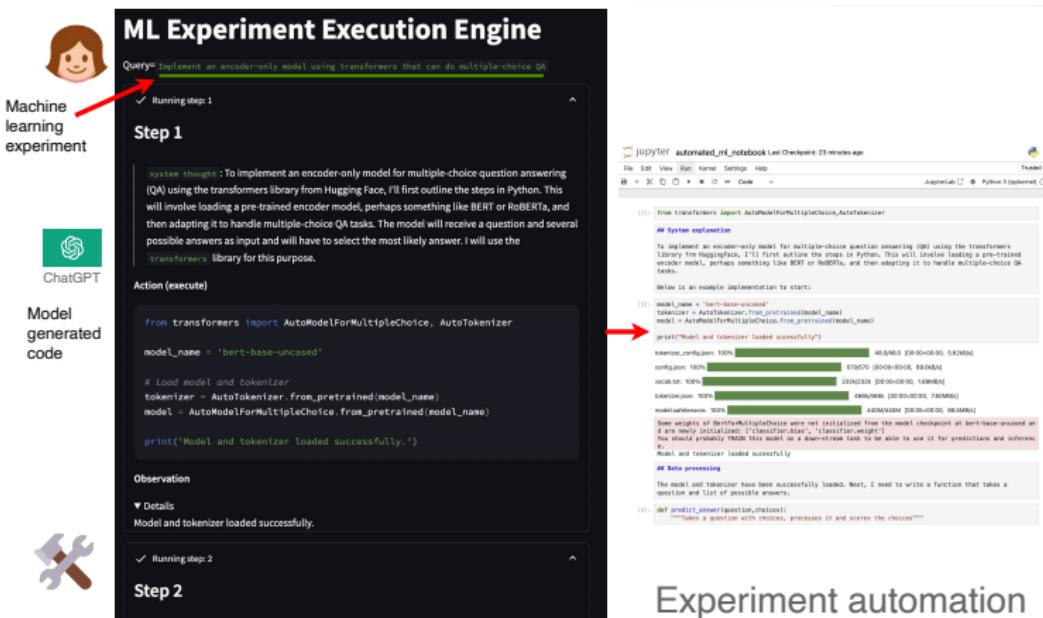
The model and tokenizer have been successfully loaded. Next, I need to write a function that takes a positive and list of possible answers.

```
def predict_(question, choices):
    # takes a question with choices, processes it and stores the choices***
```

Experiment automation

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

Language models as part of complex systems



Experiment automation

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

Language models as part of complex systems

The screenshot shows the "ML Experiment Execution Engine" interface. On the left, there's a sidebar with icons for "Machine learning experiment" (a person icon), "Model generated code" (a code editor icon), and "Experiment automation" (a gear icon). The main area has two sections:

Step 1

Summary: I'm going to implement an encoder-only model for multiple choice question answering (MCQ) using the transformers library from Hugging Face. I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to the one multiple choice task. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the `transformers` library for this purpose.

Action (execute):

```
transformers --from AutoModelForMultipleChoice --return-type=py
```

Step 2

Observation:

- Details: Model and tokenizer loaded successfully

Summary: I'm going to implement an encoder-only model for multiple choice question answering (MCQ) using the transformers library from Hugging Face. I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to the one multiple choice task. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the `transformers` library for this purpose.

Action (execute):

```
transformers --from AutoModelForMultipleChoice --return-type=py
```

Details: Model and tokenizer loaded successfully

Observation:

- Details: Model and tokenizer loaded successfully

Summary: I'm going to implement an encoder-only model for multiple choice question answering (MCQ) using the transformers library from Hugging Face. I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to the one multiple choice task. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the `transformers` library for this purpose.

Action (execute):

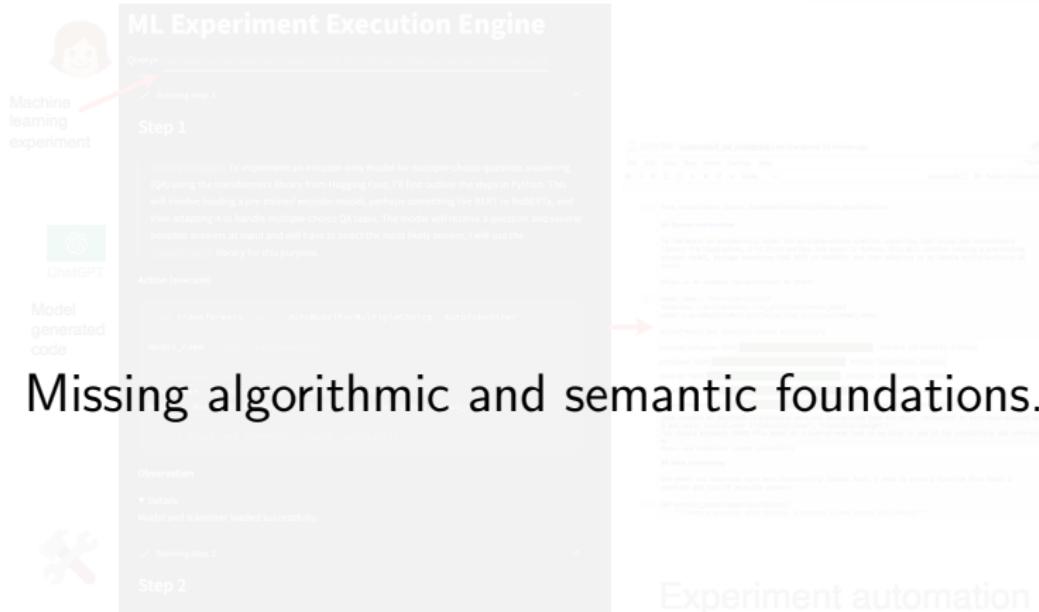
```
transformers --from AutoModelForMultipleChoice --return-type=py
```

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

Experiment automation

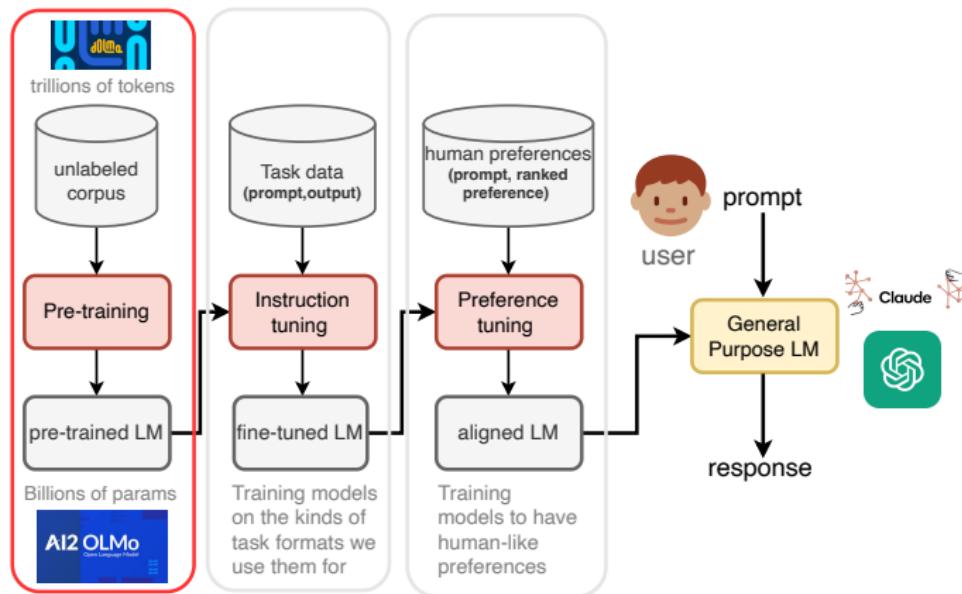
A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

Language models as part of complex systems

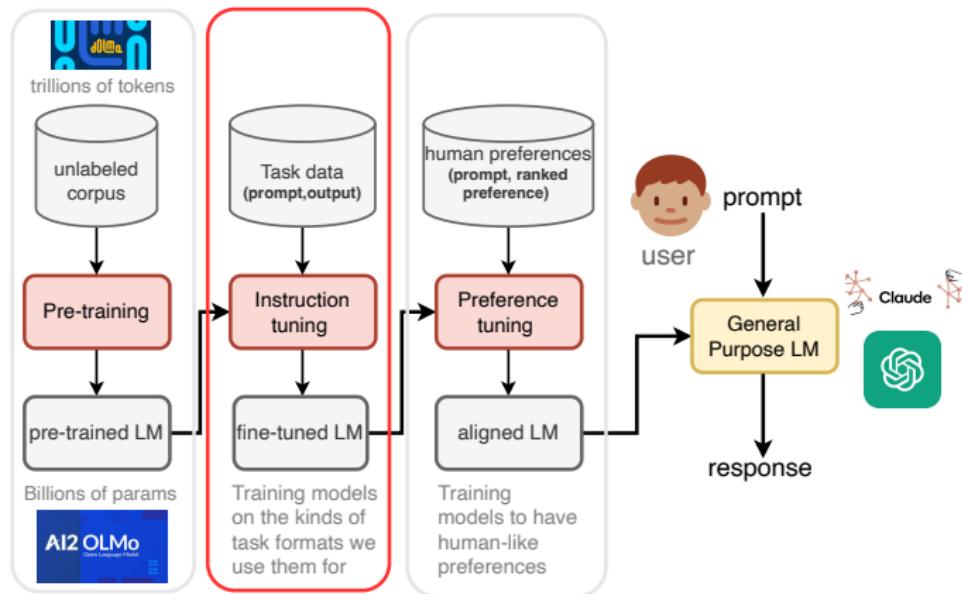


A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

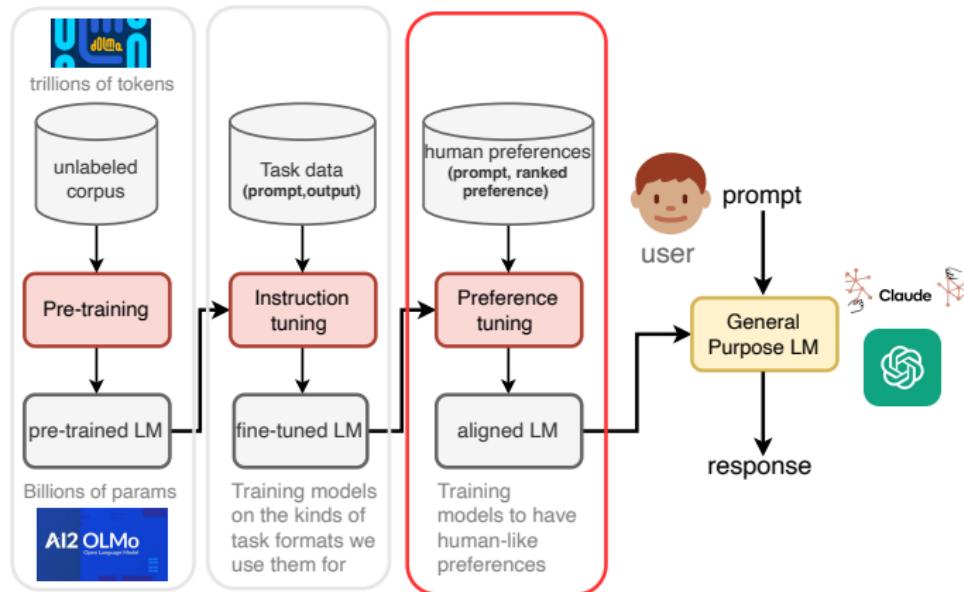
How do we get to general purpose LLMs? recipe



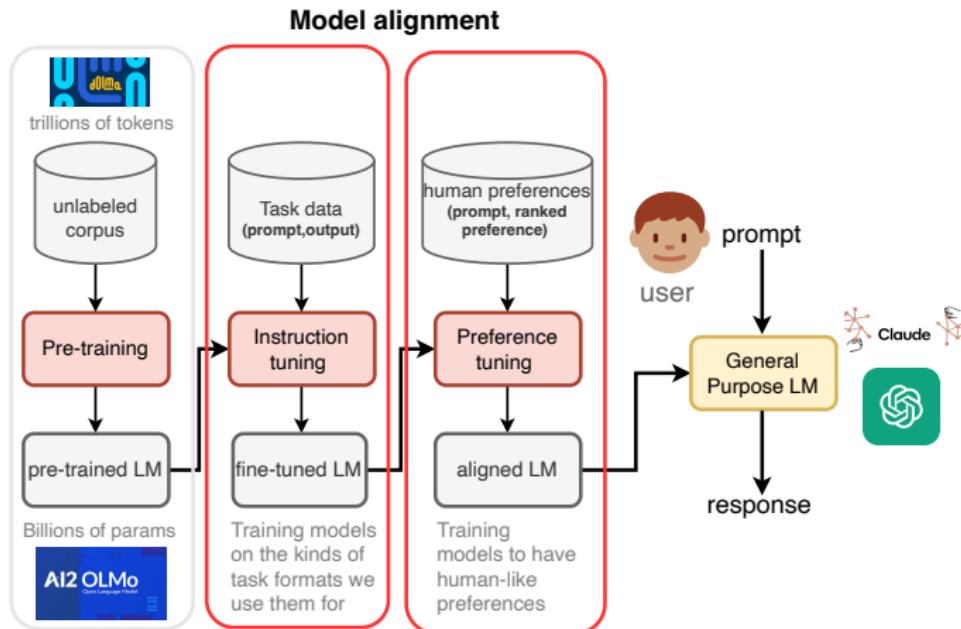
How do we get to general purpose LLMs? recipe



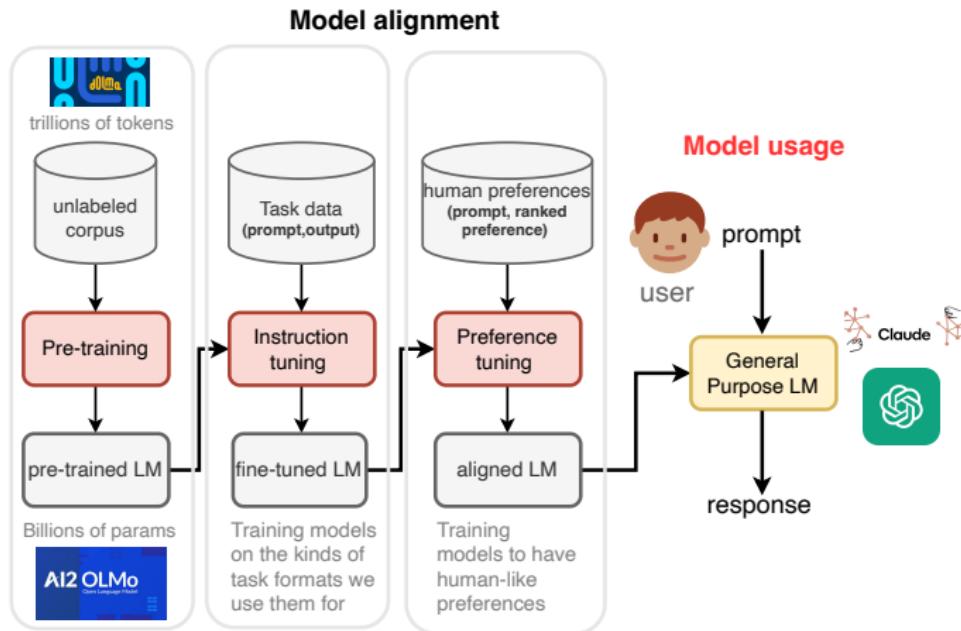
How do we get to general purpose LLMs? recipe



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

The screenshot shows the Hugging Face Model Card for the OLMo-2-1124-13B-Instruct model. The card includes the model logo, name, a note about the initial release, documentation links, and a detailed model tree.

Model tree for allenai/OLMo-2-1124-13B-Instruct:

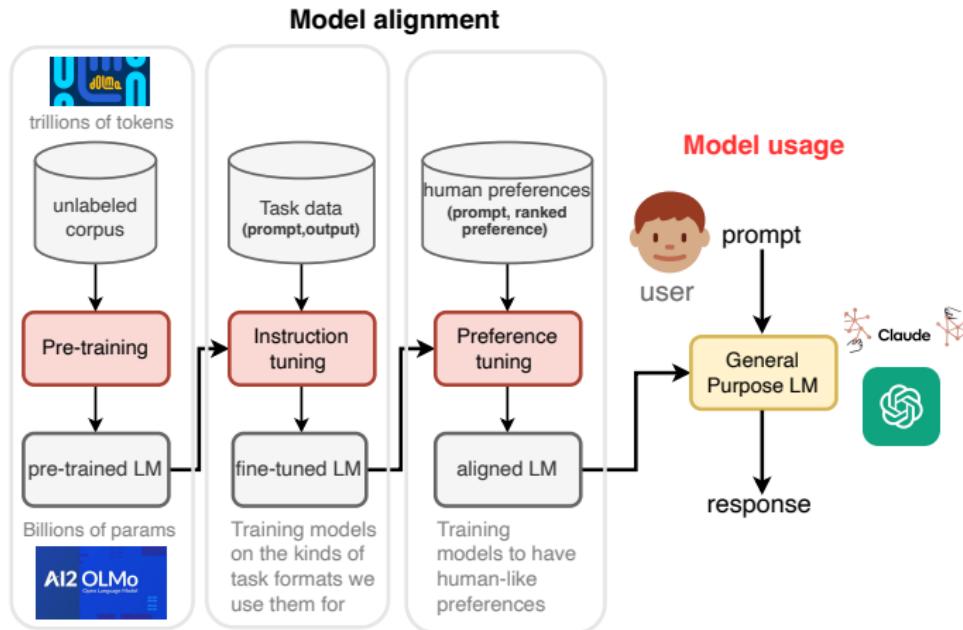
- Base model
 - Finetuned
 - Finetuned
 - Finetuned
 - Finetuned (1)
 - Adapters
 - Finetunes
 - Quantizations

Details for the Finetuned branch:
 - allenai/OLMo-2-1124-7B-SFT
 - allenai/OLMo-2-1124-7B-DPO
 - allenai/OLMo-2-1124-13B-Instruct-R1W1
 - allenai/OLMo-2-1124-13B-Instruct-R1W2

Details for the Finetuned (1) branch:
 - this model
 - 4 models
 - 2 models
 - 29 models

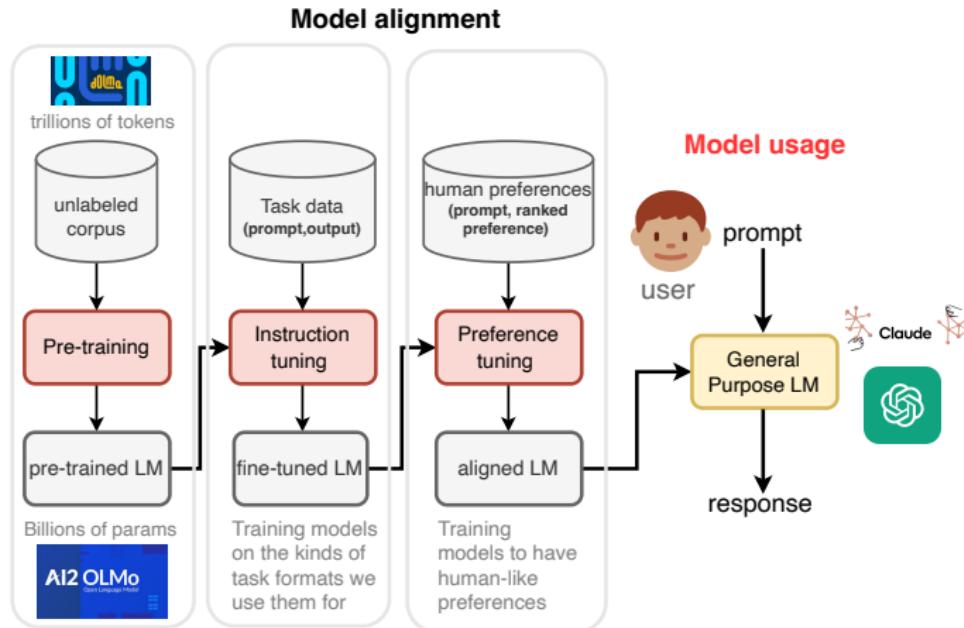
<https://allenai.org/olmo>

How do we get to general purpose LLMs? recipe



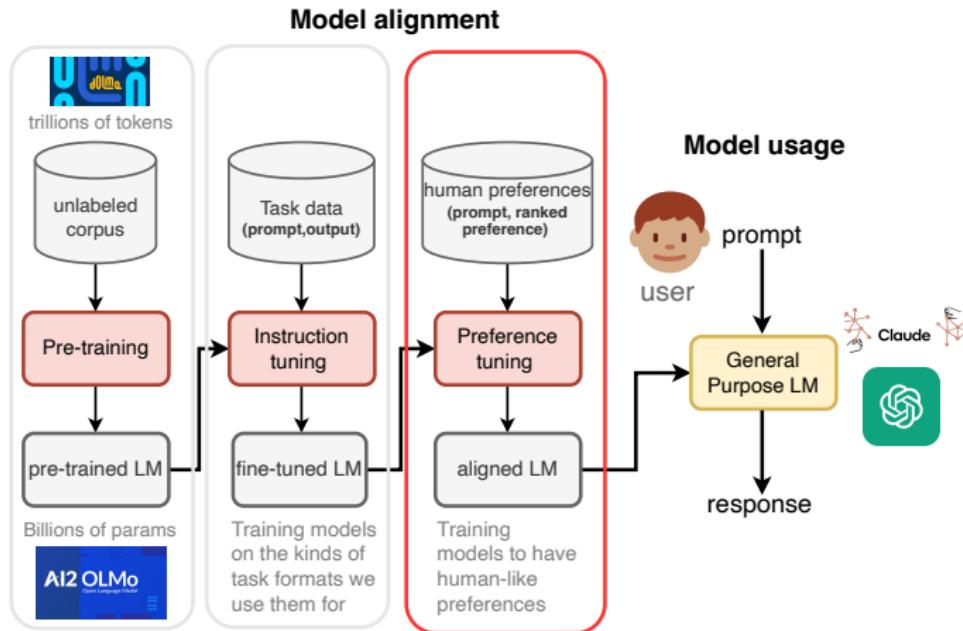
- ▶ **Dilemma:** we know vanishingly little about commercial models, models and datasets in general are huge, opaque.

How do we get to general purpose LLMs? recipe



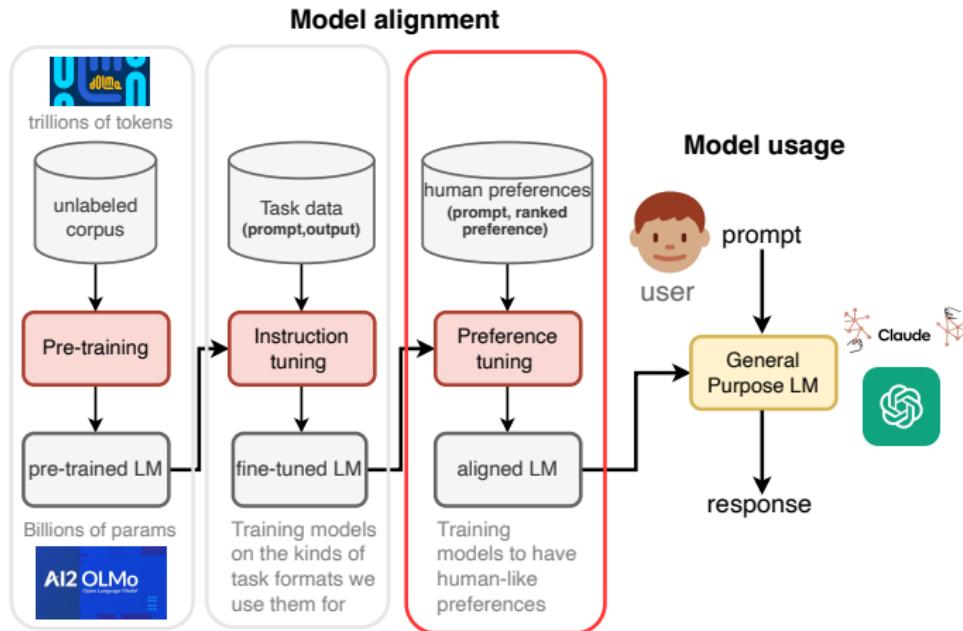
An obvious problem for safety and applications, but also for deciding what research to do, how to innovate.

Modeling the formal semantics of LLM algorithms

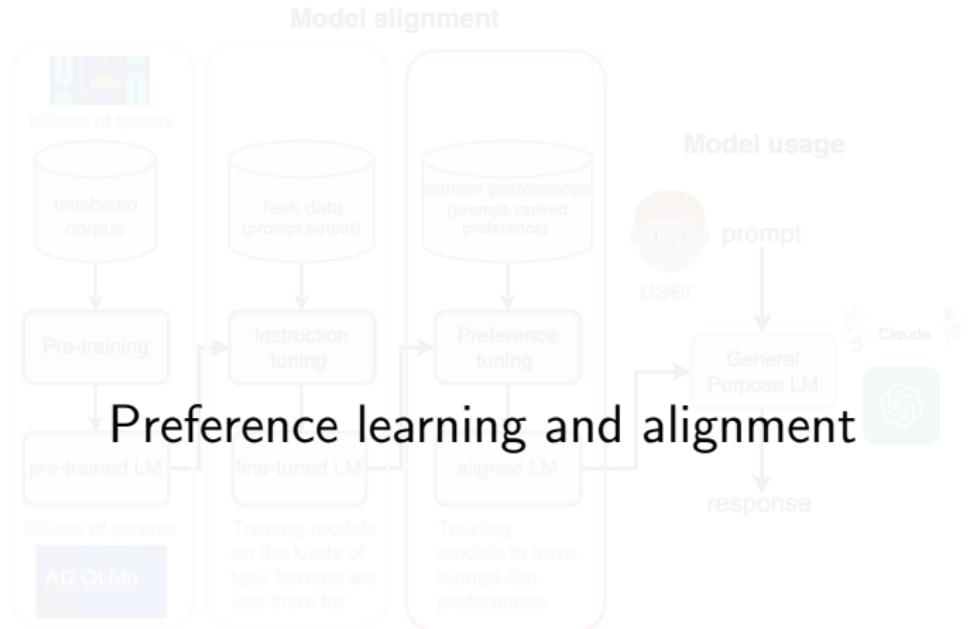


Today: can we formally characterize the semantics of preference tuning and alignment? Both for understanding and innovation; **armchair NLP**.

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?



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 :

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

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

Offline preference alignment in a nutshell

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

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

optimize a policy model $y \sim \pi_\theta(\cdot | x)$ (**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 :

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

optimize a policy model $y \sim \pi_\theta(\cdot | x)$ (**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
`{rafailev, 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^{*§}

Stefano Ermon^{*}, Michael A. Lewis[†], Chelsea Finn[†]

DPO loss function

^{*}Stanford University [†]CZ Biohub

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

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 feedback, and then the reward model is used to reward LMs for aligning their responses to the reward model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the components of a DPO loss function, which we call *Direct Preference Optimization* (DPO), and its training algorithm, which we call *Direct Preference Alignment* (DPA).

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

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^{a†}

Archit Sharma^{a*}

Eric Mitchell^{a†}

Stefano Ermon^{b†}

Christopher D. Manning^b

Chelsea Finn^b

^aStanford University ^bCZ Biohub
(rafailev,architeh,eric.mitchell)@cs.stanford.edu

These equations are not easy to understand

Abstract. Fine-tuning large unsupervised language models (LMs) to align with user’s old 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^{*§}

Stefano Ermon^{*}, Michael D. Noll[†], Chelsea Finn[†]

DPO loss function

^{*}Stanford University [†]CZ Biohub

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

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 policies are trained on, and then fine-tuning the language model. This approach is slow and requires many human annotations.

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

RLHF is a complex and often unstable procedure, first fitting a reward model that policies are trained on, and then fine-tuning the language model. This approach is slow and requires many human annotations. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy directly from the language model. Our proposed DPO loss function is the reward model in RLHF that enables extraction of the corresponding optimal policy directly from the language model. 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.

The many varieties of DPO

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

Rafael Rafailev^{*†}

Archit Sharma^{*†}

Eric Mitchell^{*‡}

Stefano Ermon^{†‡}

Christopher D. Manning[†]

Chelsea Finn[†]

^{*}Stanford University [†]CZ Biohub
`{rafailev, architsb, 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 Rafailev^{*†}

Archit Sharma^{*†}

Eric Mitchell^{*‡}

Stefano Ermon[‡]

Christopher D. Manning[†]

Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
rafailev, archits, 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]	$\max \left(0, -\frac{1}{ y_w } \log \pi_\theta(y_w x) + \frac{1}{ y_l } \log \pi_\theta(y_l x) \right) - \lambda \log \pi_\theta(y_w x)$
SLiC-HF [96]	$\max(0, \delta - \log \pi_\theta(y_w x) + \log \pi_\theta(y_l x)) - \lambda \log \pi_\theta(y_w x)$
DPO [66]	$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$
IPO [6]	$\left(\log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} - \frac{1}{2\tau} \right)^2$
CPO [88]	$-\log \sigma \left(\beta \log \pi_\theta(y_w x) - \beta \log \pi_\theta(y_l x) \right) - \lambda \log \pi_\theta(y_w x)$
KTO [29]	$-\lambda_{\text{mt}} \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - z_{\text{mt}} \right) + \lambda \sigma \left(z_{\text{mt}} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$, where $z_{\text{mt}} = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\text{KL}(\pi_\theta(y x) \pi_{\text{ref}}(y x))]$
ORPO [42]	$-\log p_\theta(y_w x) - \lambda \log \sigma \left(\log \frac{p_\theta(y_w x)}{1-p_\theta(y_w x)} - \log \frac{p_\theta(y_l x)}{1-p_\theta(y_l x)} \right)$, where $p_\theta(y x) = \exp \left(\frac{1}{ y } \log \pi_\theta(y x) \right)$
R-DPO [64]	$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} + (\alpha y_w - \alpha y_l) \right)$
SimPO	$-\log \sigma \left(\frac{\beta}{ y_w } \log \pi_\theta(y_w x) - \frac{\beta}{ y_l } \log \pi_\theta(y_l x) - \gamma \right)$

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 Rafailev^{*†}

Archit Sharma^{*†}

Eric Mitchell^{*‡}

Stefano Ermon[‡]

Christopher D. Manning[†]

Chelsea Finn[†]

[†]Stanford University [‡]CZ Biohub
rafailev,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]	$\max \left(0, -\frac{1}{ y_w } \log \pi_\theta(y_w x) + \frac{1}{ y_l } \log \pi_\theta(y_l x) \right) - \lambda \log \pi_\theta(y_w x)$
SLiC-HF [96]	$\max(0, \delta - \log \pi_\theta(y_w x) + \log \pi_\theta(y_l x)) - \lambda \log \pi_\theta(y_w x)$
DPO [66]	$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$
IPO [6]	$\left(\log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} - \frac{1}{2\tau} \right)^2$
CPO [88]	$-\log \sigma \left(\beta \log \pi_\theta(y_w x) - \beta \log \pi_\theta(y_l x) \right) - \lambda \log \pi_\theta(y_w x)$
KTO [29]	$-\lambda_{\text{mt}} \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - z_{\text{mt}} \right) + \lambda \sigma \left(z_{\text{mt}} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$, where $z_{\text{mt}} = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\mathcal{BKL}(\pi_\theta(y x), \pi_{\text{ref}}(y x))]$
ORPO [42]	$-\log p_\theta(y_w x) - \lambda \log \sigma \left(\log \frac{p_\theta(y_w x)}{1-p_\theta(y_w x)} - \log \frac{p_\theta(y_l x)}{1-p_\theta(y_l x)} \right)$, where $p_\theta(y x) = \exp \left(\frac{1}{ y } \log \pi_\theta(y x) \right)$
R-DPO [64]	$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_\theta(y_l x)}{\pi_{\text{ref}}(y_l x)} + (\alpha y_w - \alpha y_l) \right)$
SimPO	$-\log \sigma \left(\frac{\beta}{ y_w } \log \pi_\theta(y_w x) - \frac{\beta}{ y_l } \log \pi_\theta(y_l x) - \gamma \right)$

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

Robert Rothlauf¹ · Austin Stevens² · Eric Michail³
Sethuraman Kannan⁴ · Christopher D. Manning⁵ · Clinton Fife⁶

Stanford University TREC NLP Group
https://nlp.stanford.edu/pubs/2023_trec.pdf

While language models can be used to estimate user satisfaction, this is often difficult due to the complex nature of what makes for good and satisfying text. In this paper, we propose a new approach to estimating user satisfaction based on direct preference optimization (DPO). DPO is a generalization of reinforcement learning that allows us to directly optimize for user satisfaction without requiring a reward function. We show that DPO can be used to estimate user satisfaction for a variety of tasks, including text generation, text summarization, and text classification. We also show that DPO can be used to estimate user satisfaction for a variety of tasks, including text generation, text summarization, and text classification.

Why this can be frustrating

DPO loss

$$-\log \sigma \left(\beta \log \frac{\pi_\theta(a|s)}{\pi_\theta(a'|s)} - \beta \log \frac{\pi_\theta(a|s)}{\pi_\theta(a'|s)} \right)$$

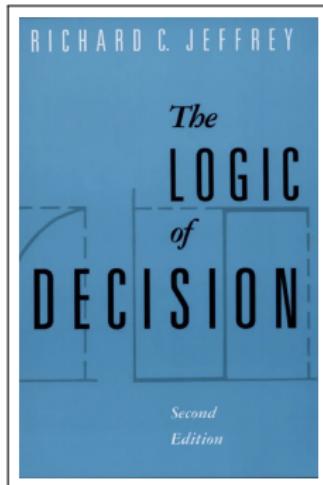
DPO variants

DPO	$\log \pi_\theta(a s)$	(softmax distribution) + baseline
CPO (DPO)	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline
KTDPO	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline
CDPO (DPO)	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline
KL-DPO (DPO)	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline
RL-DPO (DPO)	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline
DDPO	$\log \pi_\theta(a s) - \log \pi_\theta(a' s)$	(softmax distribution) + baseline

from Meng et al. (2024)

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)

Preference Principle	THE STATUS OF VARIOUS PREFERENCE PRINCIPLES					
	Von Wright	Chisholm Sosa	Martin	P*	P★	
1. $pPq \rightarrow \sim(qPp)$	✓	✓	✓	+	+	+
2. $(pPq \& qPr) \rightarrow pPr$	✓	✓	✓	+	+	+
3. $pPq \rightarrow \sim qP \sim p$		x	✓	(+) ¹	+	+
4. $\sim qP \sim p \rightarrow pPq$		x	✓	(+) ¹	+	+
5. $pPq \rightarrow (p \& \sim q) P(\sim p \& q)$	✓	✓		+	+	+
6. $(p \& \sim q) P(\sim p \& q) \rightarrow pPq$	✓	x		+	+	+
7. $[\sim(pP \sim p) \& \sim(\sim pPp) \& \sim(qP \sim q) \& \sim(\sim qPq)] \rightarrow [\sim(pPq) \& \sim(qPp)]$	✓	✓		+	+	+
8. $[\sim(qP \sim q) \& \sim(\sim qPq) \& pPq] \rightarrow pP \sim p$	✓	✓		+	+	-
9. $[\sim(qP \sim q) \& \sim(\sim qPq) \& qP \sim p] \rightarrow pP \sim p$	✓	✓		+	+	-
10. $pPq \rightarrow [(p \& r) P(q \& r) \& (p \& \sim r)]$	✓			-	-	+
11. $[(p \& r) P(q \& r) \& (p \& \sim r) P(q \& \sim r)] \rightarrow pPq$	✓			(+) ²	(+) ³	+
12. $[\sim(pPq) \& \sim(qPr)] \rightarrow \sim(pPr)$		✓		+	+	-
13. $(pPr \vee qPr) \rightarrow (p \vee q) Pr$			✓	-	-	-
14. $(p \vee q) Pr \rightarrow [pPr \& qPr]$	✓			-	-	-
15. $[pPr \& qPr] \rightarrow (p \vee q) Pr$	✓			-	-	-
16. $(p \vee q) Pr \rightarrow (pPr \vee qPr)$				-	-	-
17. $pP(g \vee r) \rightarrow (pPp \& pPr)$	✓			-	-	-
18. $(pPq \& pPr) \rightarrow pP(q \vee r)$				-	-	-
19. $(pPr \& qPr) \rightarrow (p \& q) Pr$				-	-	-

Semantic foundations for the logic of preference Rescher (1967)

The language of machine learning

Loss functions

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

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

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



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

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



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.

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

The language of machine learning

Loss functions

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



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

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



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

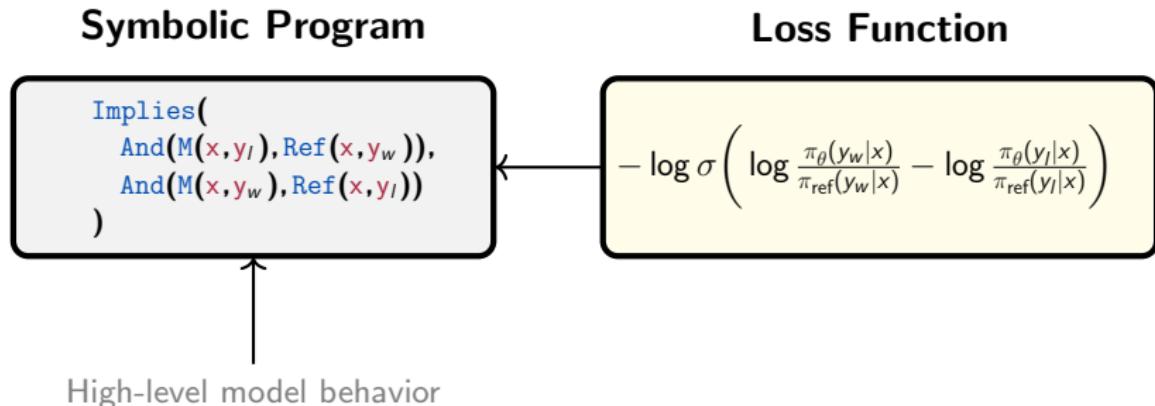
Loss Function

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

Two models, four predictions

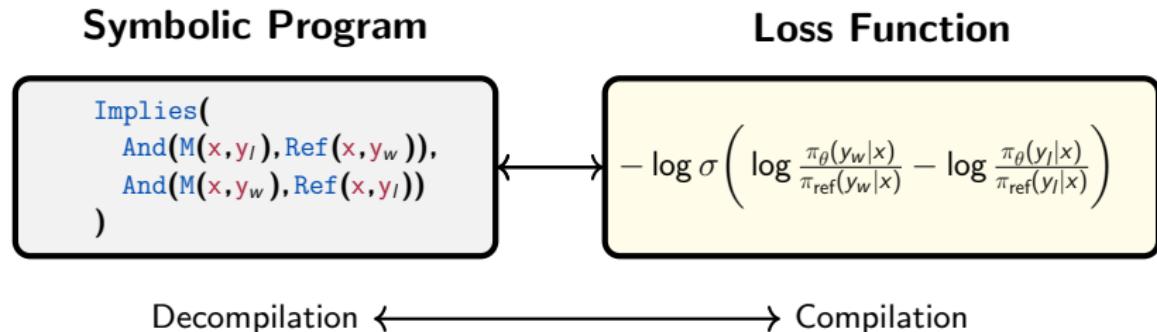
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



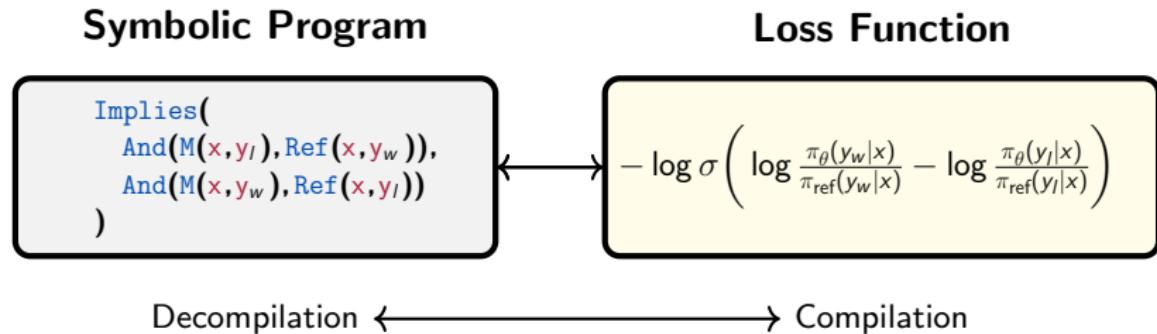
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



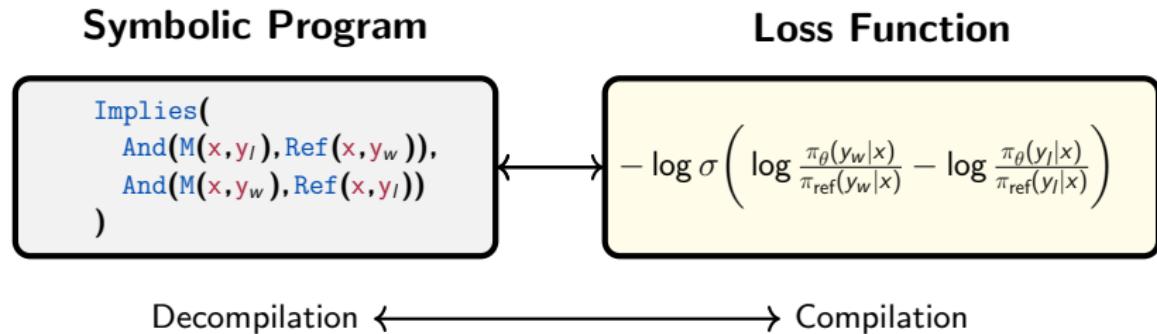
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



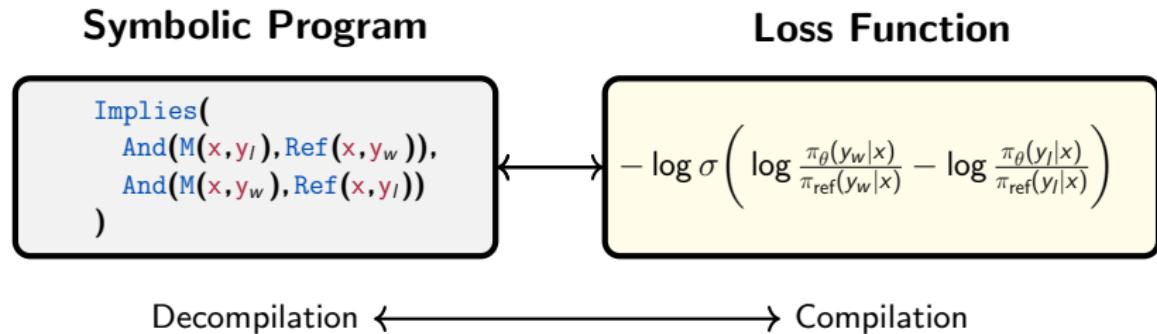
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



- ▶ **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



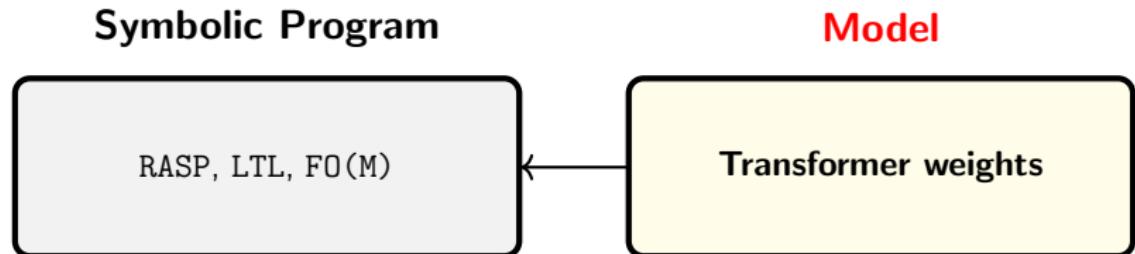
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?
 1. **Compilation:** Translating specifications into loss, well studied.
 2. **Decompilation:** Losses to specifications (inverse), less explored.

Formal analysis via decompilation in general

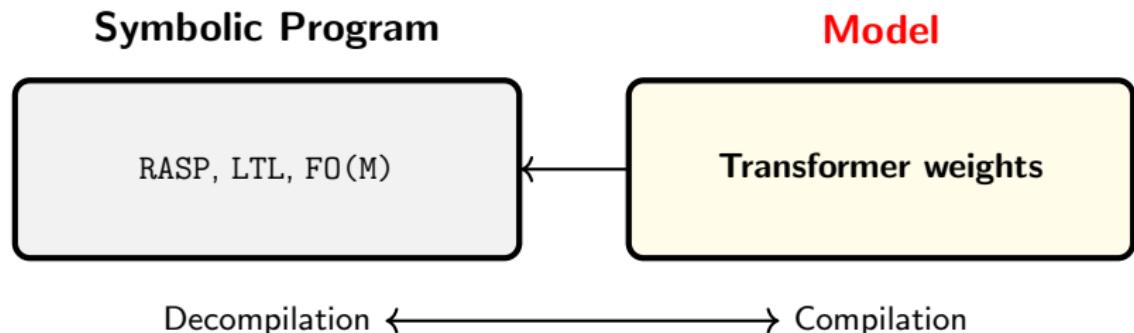
Model

Transformer weights

Formal analysis via decompilation in general



Formal analysis via decompilation in general



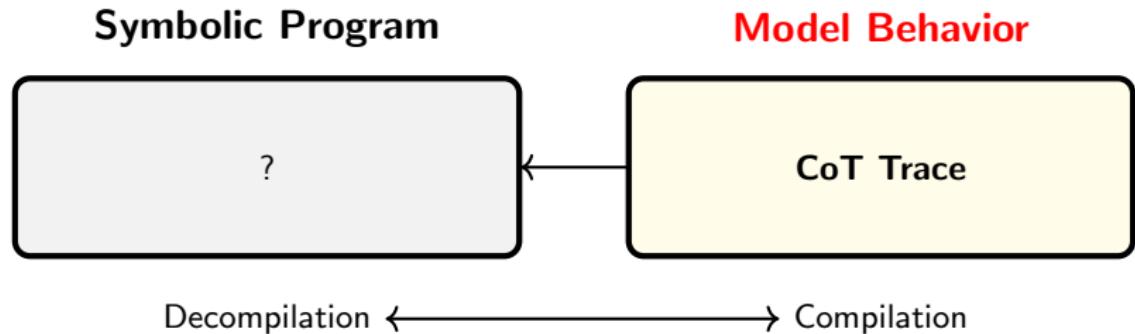
- We know what the *target languages* are ([Weiss et al., 2021](#); [Merrill and Sabharwal, 2023](#); [Yang and Chiang, 2024](#)), how to compile, decompile ([Friedman et al., 2023](#)).

Formal analysis via decompilation in general

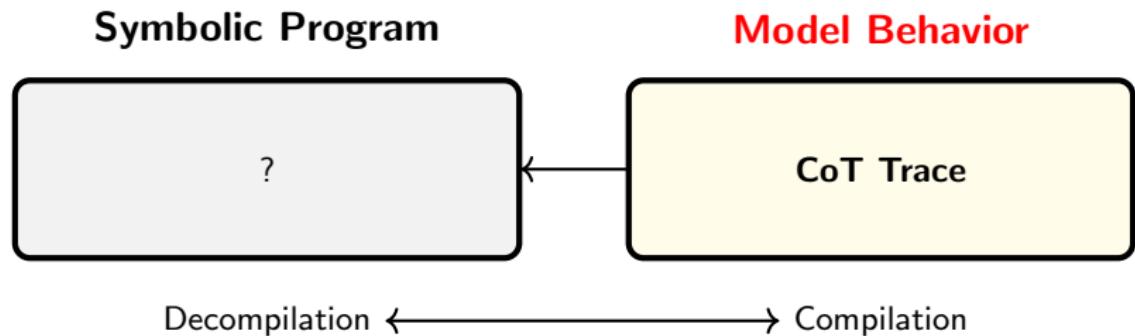
Model Behavior

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](#), [limgl](#), [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>

Language model programming: ESSLLI 2025

Lecturers

Hao Heinecke (Allen Institute for AI)

Günther Wärthen (Leiden Institute of Advanced Computer Science)

Slides

[lecture 1: course overview, language modeling basics, transformers, DCCP](#)

[lecture 2: declarative approaches to model training and fine-tuning, the ~~semantic loss~~ and ~~weighted model averaging~~, other approaches.](#)

[lecture 3: high-level programming techniques for direct preference alignment and LLM alignment, formal characterizations of known loss functions.](#)

[lecture 4: distributed inference, distributed training, fast-time inference, LLM and preference consistency, defining LLMs as programs.](#)

What is the right programming language for preference?

background: [probabilistic programming](#), [transformers](#)

extra lectures: [Prompting as programming](#), [Grammar-constrained decoding](#)

Helpful Resources

Below are some pointers to code resources:

- [languages](#) [[Julia](#), [Python](#), [JavaScript](#), [Perl](#), [Java](#), [NumPy](#)], [interpreters](#)
- [automated reasoning tools/circuits](#) [[Erlang](#), [Prolog](#), [SMT](#), [SAT](#)]
- [NLP and general ML](#) [[Transformers](#), [PyTorch](#), [TensorFlow](#), [PyTorch](#), [TensorFlow](#)]
- [other useful utilities](#) [[Jupyter](#)]

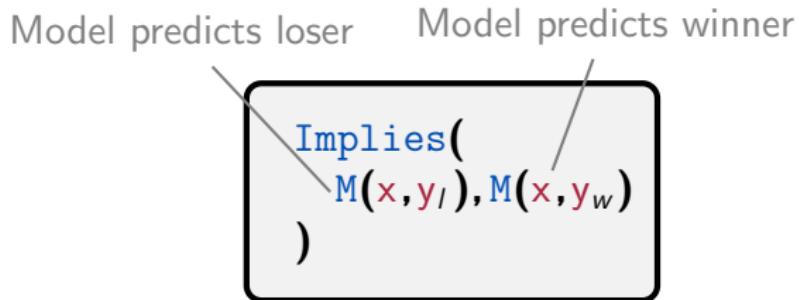
Useful tutorials: [Transformers from scratch](#) (some examples/code used in Lecture 2), [Lectures on Probabilistic Programming](#), [Probabilistic Models](#)

<https://github.com/yakazimir/LMProgramming>

Declarative models of preference

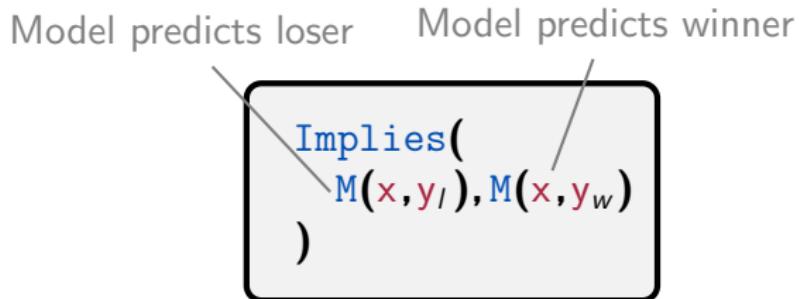
```
Implies(  
    M(x,yl), M(x,yw)  
)
```

Declarative models of preference



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

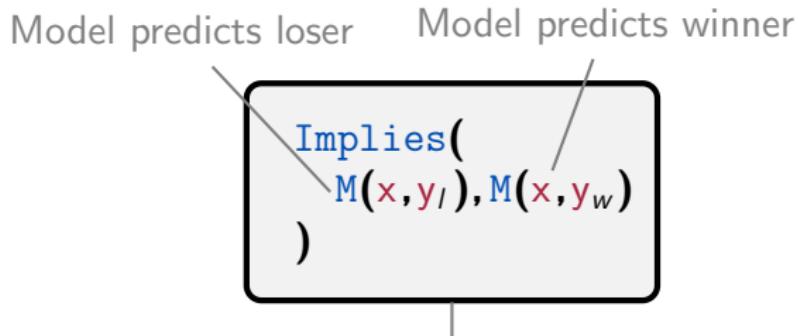
Declarative models of preference



$$w(M(x, y)) = \pi_M(y | x)$$

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

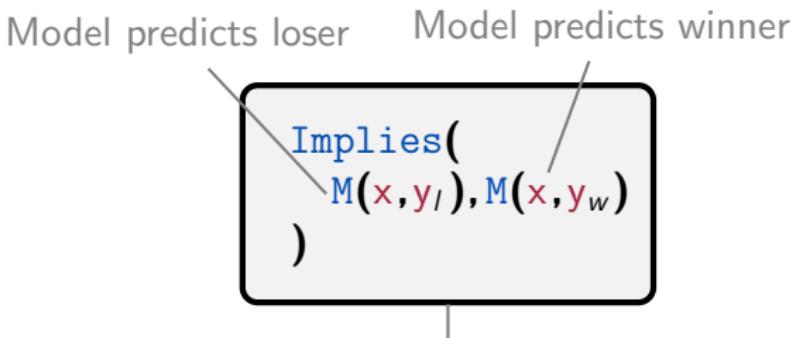
Declarative models of preference



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



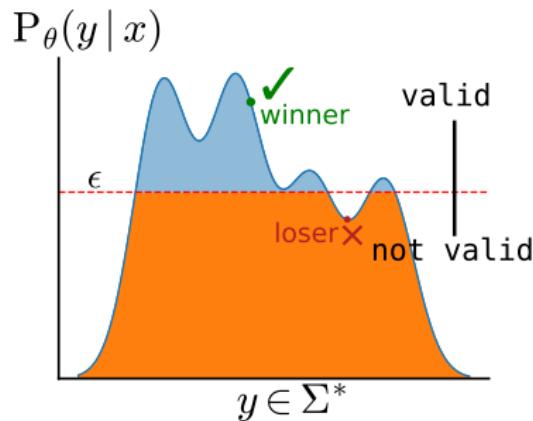
Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

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.

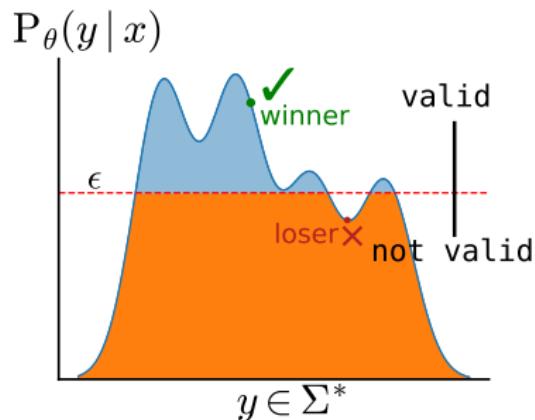


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

Uncovering the natural logic of these algorithms

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

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

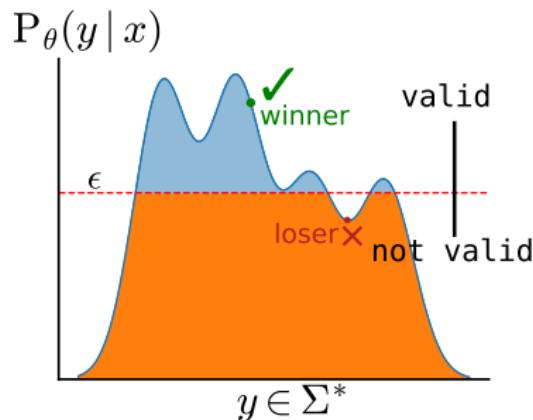


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

Uncovering the natural logic of these algorithms

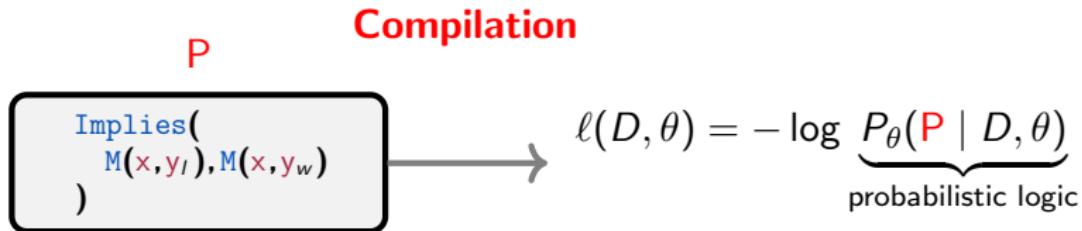
Implies(
M(x, y_l), M(x, y_w))

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



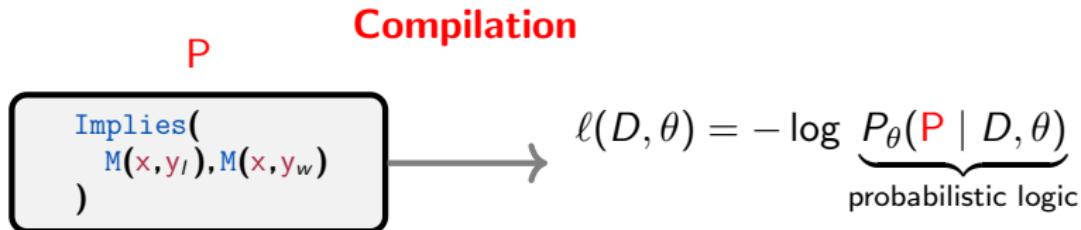
Observation: The second program is more strict than the first, involves semantic entailment.

Compilation and decompilation again



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

Compilation and decompilation again



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(

$M(x, y_l), M(x, y_w)$

)

$\ell_{\text{CP0}} = - \log \sigma \left(\log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)} \right)$

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(
 $M(x, y_l), M(x, y_w)$
)

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

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

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

Compilation and decompilation again

Decompilation

P

$\text{Implies}\left(\text{M}(x, y_l), \text{M}(x, y_w)\right)$

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

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

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

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 ℓ_{ORPO}

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

Illustration of approach and results

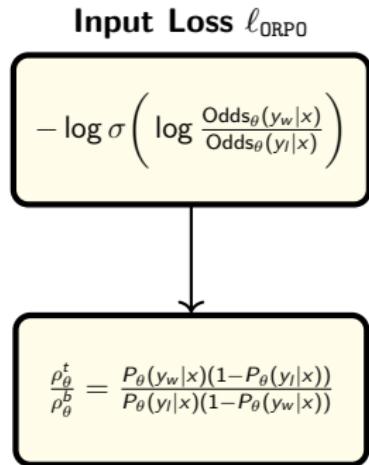


Illustration of approach and results

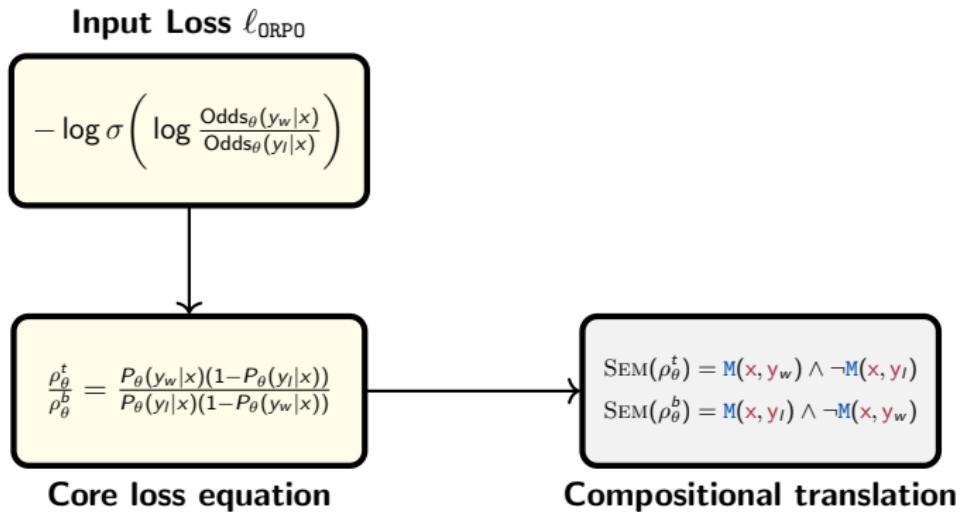


Illustration of approach and results

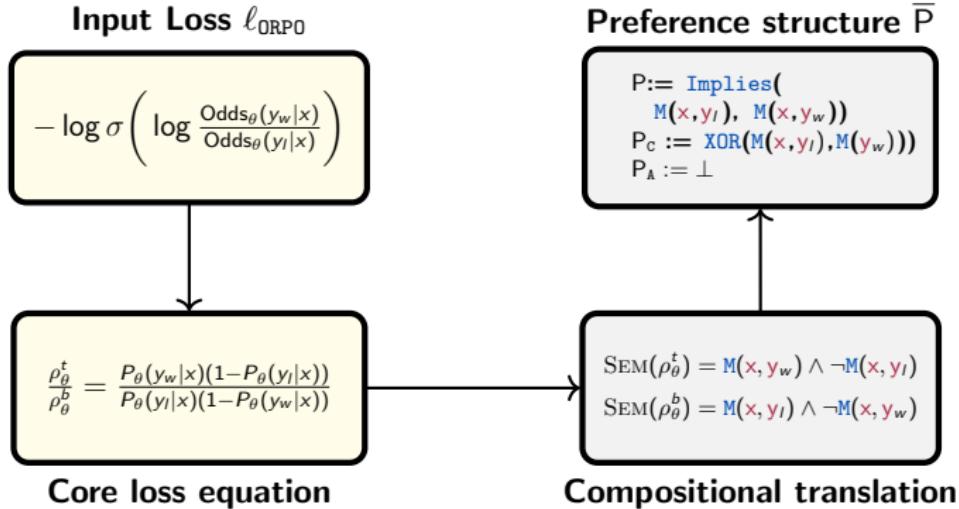


Illustration of approach and results

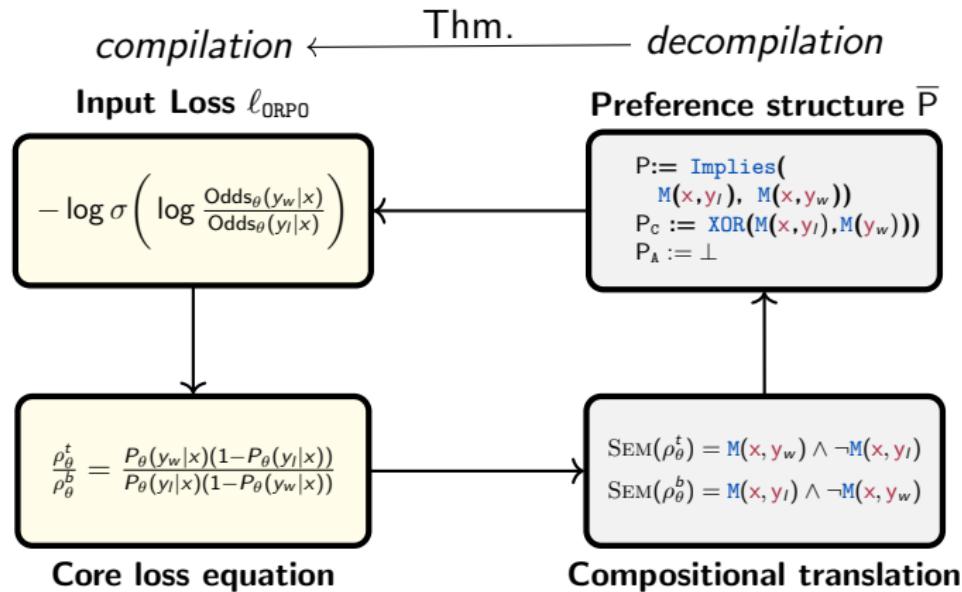
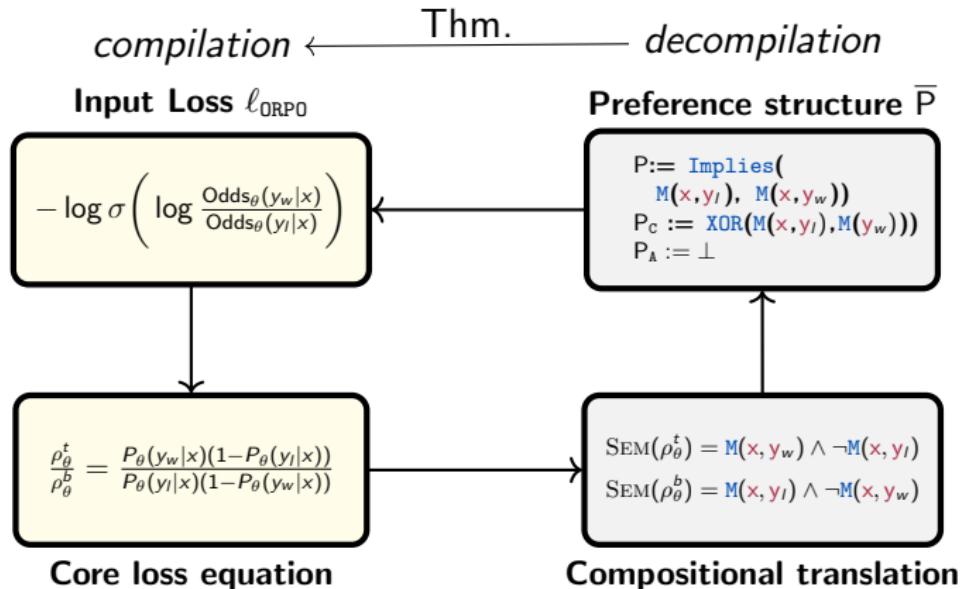
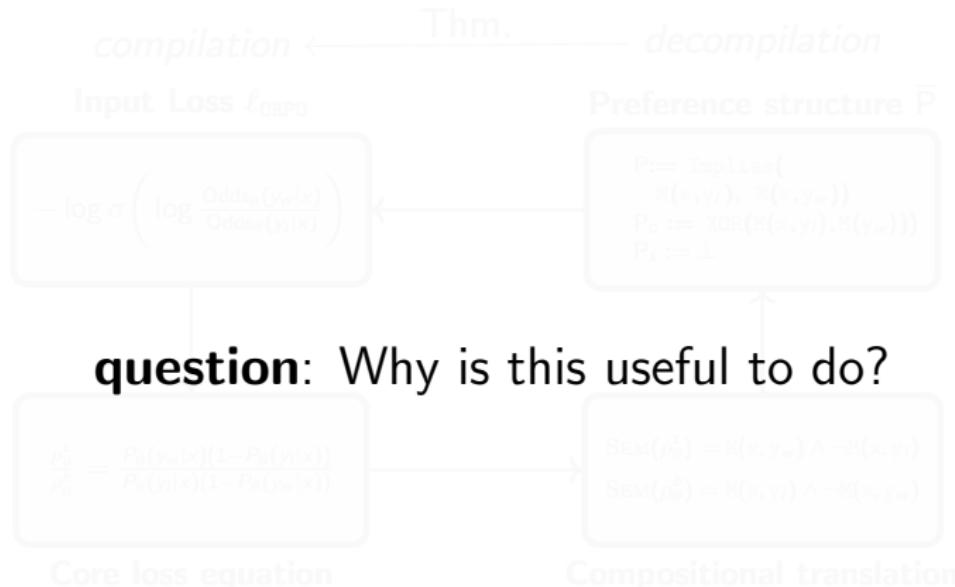


Illustration of approach and results



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

Illustration of approach and results



- ▶ 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(
 $M(x, y_I), M(x, y_w)$
)

$P^{(2)}$

And(
 $M(x, y_w),$
 Not($M(x, y_I)$)))

Boolean functions, 2 variables

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

Why is this useful? understanding the space

Boolean functions, 2 variables

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

$P^{(1)}$

$P^{(2)}$

Implies(
 $M(x, y_I), M(x, y_w)$)

And(
 $M(x, y_w),$
 $\text{Not}(M(x, y_I)))$

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

Why is this useful? understanding the space

Boolean functions, 2 variables

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

$P^{(1)}$

$\text{Implies}(\text{M}(x, y_I), \text{M}(x, y_w))$

$P^{(2)}$

$\text{And}(\text{M}(x, y_w), \text{Not}(\text{M}(x, y_I)))$

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

Why is this useful? understanding the space

Boolean functions, 2 variables

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

$\text{Implies}(M(x, y_I), M(x, y_w))$

$\text{And}(M(x, y_w), \text{Not}(M(x, y_I)))$

no reference: 256 losses

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

Loss functions as truth tables

Implies(
And($M(x, y_I)$, $Ref(x, y_w)$),
And($M(x, y_w)$, $Ref(x, y_I)$))
)

4 variables

$Ref(x, y_w)$	$M(x, y_I)$	$Ref(x, y_I)$	$M(x, y_w)$
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($M(x, y_I)$, Ref(x, y_w)),
And($M(x, y_w)$, Ref(x, y_I)))

)

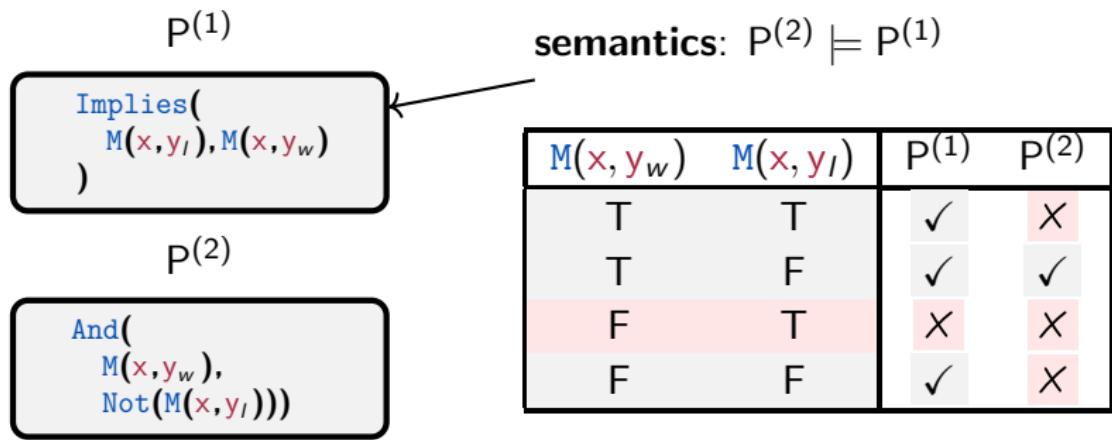
4 variables

answer: loads.

Ref(x, y_w)	$M(x, y_I)$	Ref(x, y_I)	$M(x, y_w)$
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
F	T	T	T
T	F	F	F
T	F	T	F
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

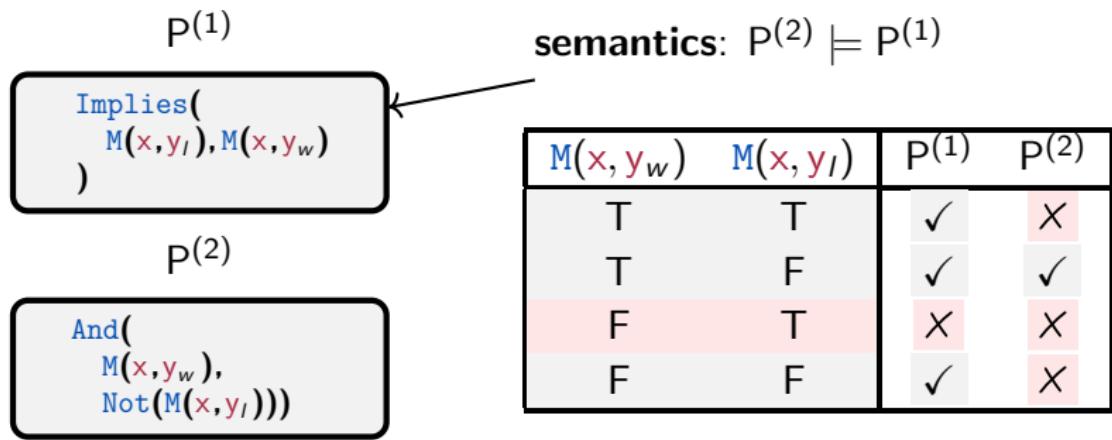
w/ reference: 4,294,967,296 losses

Why is this useful? understanding the structure



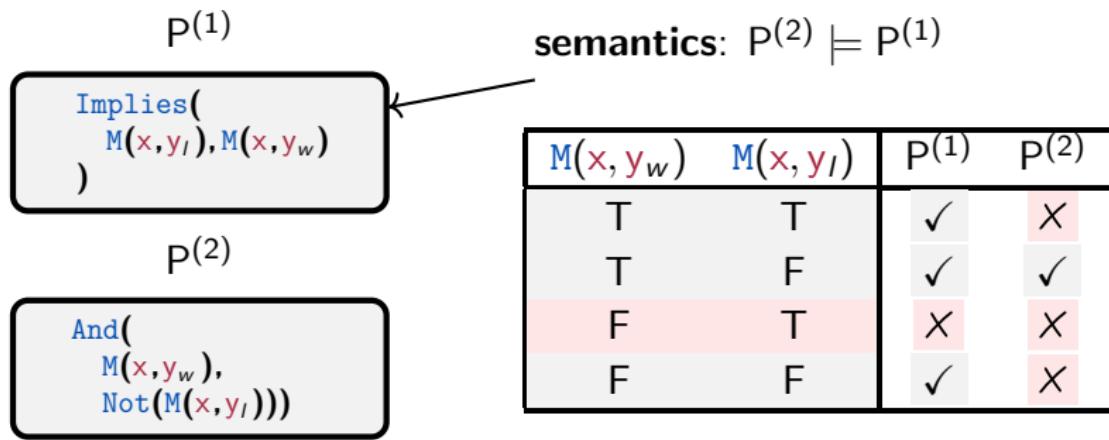
Proposition (Xu et al., 2018): Loss behavior is monotonic w.r.t semantic entailment: if $P^{(2)} \models P^{(1)}$ then $\ell(D, \theta, P^{(2)}) \geq \ell(D, \theta, P^{(1)})$.

Why is this useful? understanding the structure



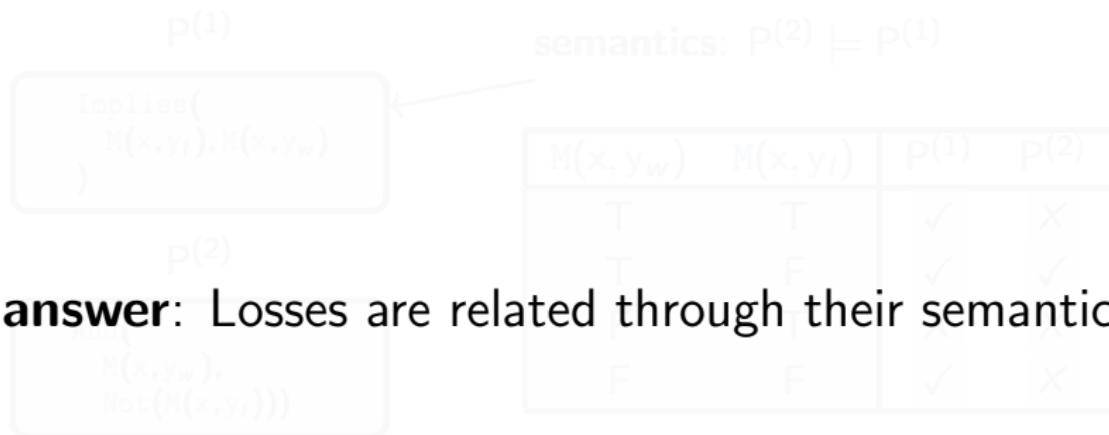
Proposition (Xu et al., 2018): Loss is equivalent under semantic equivalence: If $P^{(2)} \equiv P^{(1)}$ then $\ell(D, \theta, P^{(2)}) = \ell(D, \theta, P^{(1)})$.

Why is this useful? understanding the structure



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

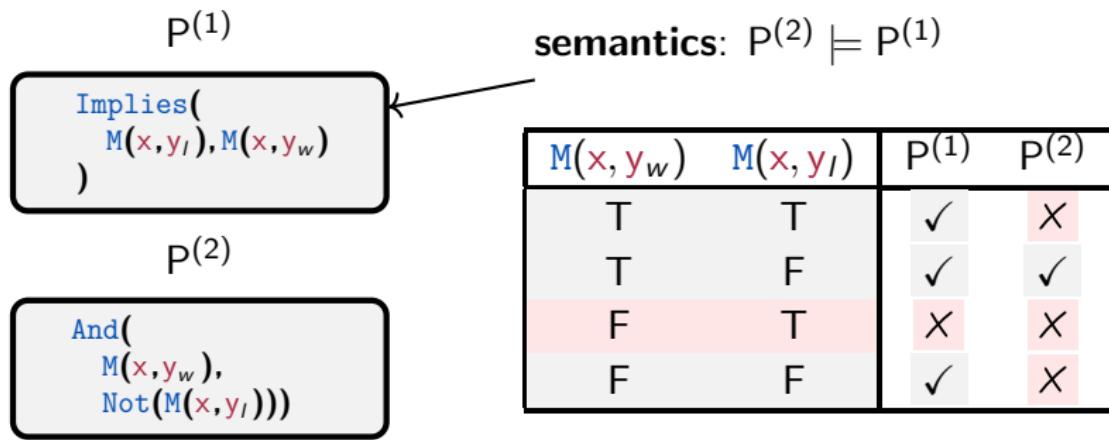
Why is this useful? understanding the structure



answer: Losses are related through their semantics

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



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,yI), Ref(x,yw)),  
    And(M(x,yw), Ref(x,yI))  
)
```

DPO Loss

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

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(  
    And(M(x,yI), Ref(x,yw)),  
    And(M(x,yw), Ref(x,yI))  
)
```

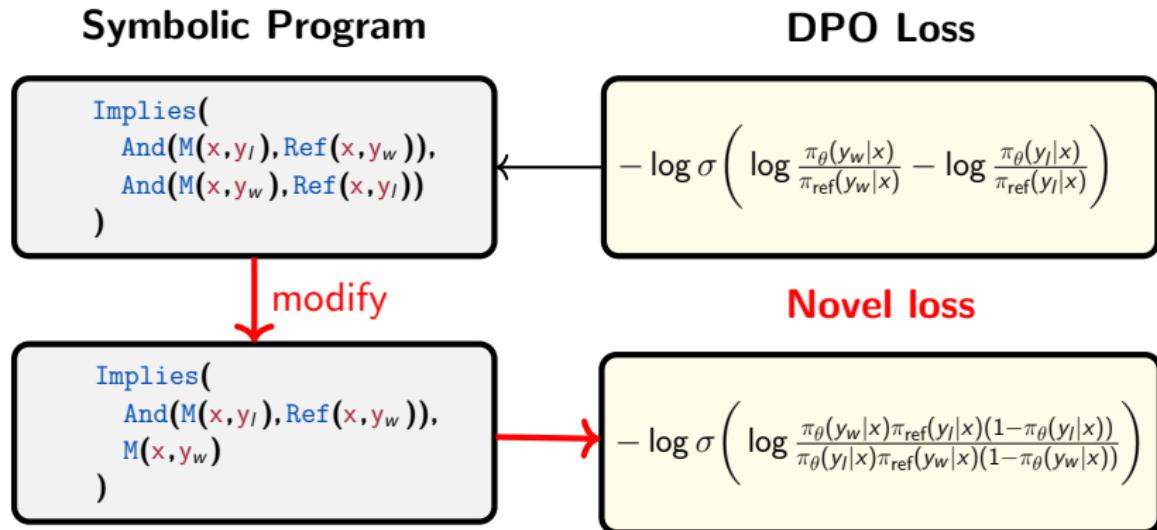
DPO Loss

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

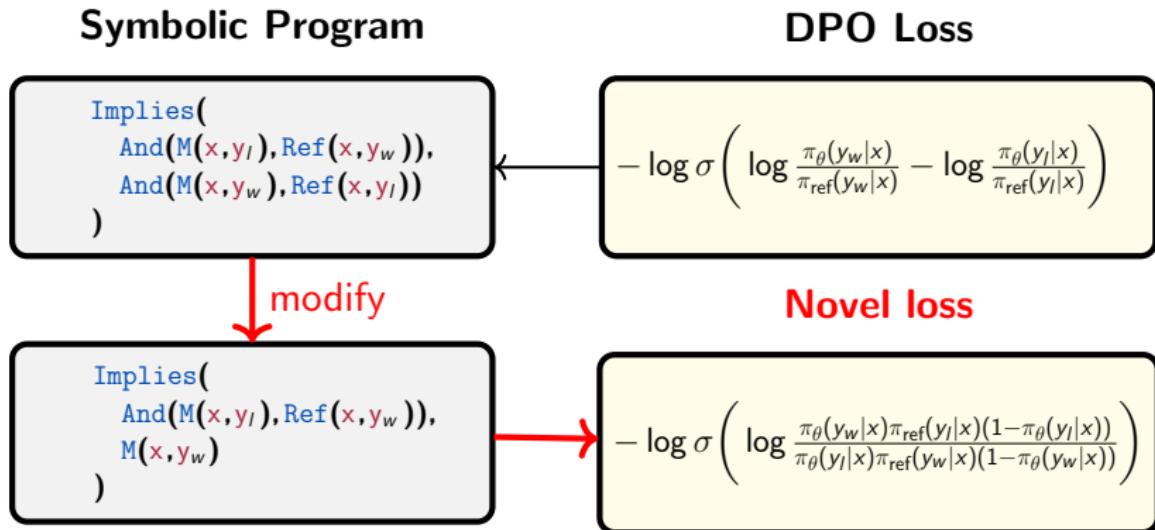
modify


```
Implies(  
    And(M(x,yI), Ref(x,yw)),  
    M(x,yw)  
)
```

Deriving new losses symbolically, from first principles



Deriving new losses symbolically, from first principles



- ▶ High-level programming language for defining new losses.

Deriving new losses symbolically, from first principles

Symbolic Program

```
Implies(  
    And(M(x,yI), Ref(x,yw)),  
    And(M(x,yw), Ref(x,yI))  
)
```

DPO Loss

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

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

\downarrow

```
Implies(  
    And(M(x,yI), Ref(x,yw)),  
    M(x,yw)  
)
```

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

How does the logic work? compilation

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

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

Implies(
 $M(x, y_I), M(x, y_w)$
)

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P
Implies(
 $M(x, y_I)$, $M(x, y_w)$)

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

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

- ▶ 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

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P
Implies(
 $M(x, y_I)$, $M(x, y_w)$)

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

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

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	Implies($M(x, y_l), M(x, y_w)$)
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

$$\begin{aligned}
 \underbrace{\ell_x}_{\text{column}} &:= -\log \sigma \left(\log \frac{\sum_w \text{✓}_w}{\sum_w \text{X}_w} \right) \\
 &= -\log \sigma \left(\log \frac{\pi_\theta(y_w | x)(1 - \pi_\theta(y_l | x))}{\pi_\theta(y_l | x)(1 - \pi_\theta(y_w | x))} \right) \\
 &\quad \underbrace{\ell_{\text{ORPO}}, P_\theta(\mathbf{P}|\text{one hot})}_{\ell_{\text{ORPO}}, P_\theta(\mathbf{P}|\text{one hot})}
 \end{aligned}$$

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

Implies(
 $M(x, y_I), M(x, y_w)$
)

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

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

$$= -\log \sigma \left(\log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_I | x)} \right)$$

$\underbrace{\ell_{\text{CPO}}, \sim P_\theta(\text{P} | \text{one true})}_{\ell_{\text{CPO}}, \sim P_\theta(\text{P} | \text{one true})}$

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	Implies($M(x, y_I), M(x, y_w)$)
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

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

$$w \models l(x, y)$$

$$w \not\models \neg l(x, y)$$

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

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

How does the logic work? compilation

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

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.

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

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

How does the logic work? compilation

		CPO	ORPO	unCPO	P
$M(x, y_w)$	$M(x, y_I)$				Implies($M(x, y_I), M(x, y_w)$)
T	T	✓ X		✓	
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

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

$$\bar{P} := \left(\underbrace{P}_{\text{core}}, \underbrace{P_C, P_A}_{\text{constraints}} \right)$$

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	Implies($M(x, y_I), M(x, y_w)$)
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

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

$$\{ \text{✓}, \text{X} \}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

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

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

How does the logic work? compilation

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X			
T	F	✓	✓		
F	T	X	X		
F	F			✓ ✓ X ✓	Implies($M(x, y_I), M(x, y_w)$)

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

{ note: $M(x, y_I) \rightarrow M(x, y_w) \equiv \neg M(x, y_I) \vee M(x, y_w)$

$$\begin{aligned}
 \ell_x &:= -\log \sigma \left(\log \frac{\sum_{w \vdash l(x,y)} \checkmark_w}{\sum_{w \nvdash l(x,y)} X_w} \right) \\
 &= -\log \sigma \underbrace{\left(\log \frac{\pi_\theta(y_I | x) \pi_\theta(y_w | x) + (1 - \pi_\theta(y_I | x))}{\pi_\theta(y_I | x) (1 - \pi_\theta(y_w | x))} \right)}_{\text{novel loss without constraints, } P_\theta(P|T)}
 \end{aligned}$$

What properties do real losses have?

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	Implies($M(x, y_I), M(x, y_w)$)
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

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

$$\{\checkmark, \text{X}\}_w := \prod_{w \models M(x,y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x,y)} 1 - \pi_\theta(y | x)$$

$$\begin{aligned}
 \underbrace{\ell_x}_{\text{column}} &:= -\log \sigma \left(\log \frac{\sum \checkmark_w}{\sum \text{X}_w} \right) \\
 &= -\log \sigma \left(\log \underbrace{\frac{\pi_\theta(y_I | x) \pi_\theta(y_w | x) + (1 - \pi_\theta(y_I | x))}{\pi_\theta(y_I | x) (1 - \pi_\theta(y_w | x))}}_{\text{novel loss without constraints, } P_\theta(P | \top)} \right)
 \end{aligned}$$

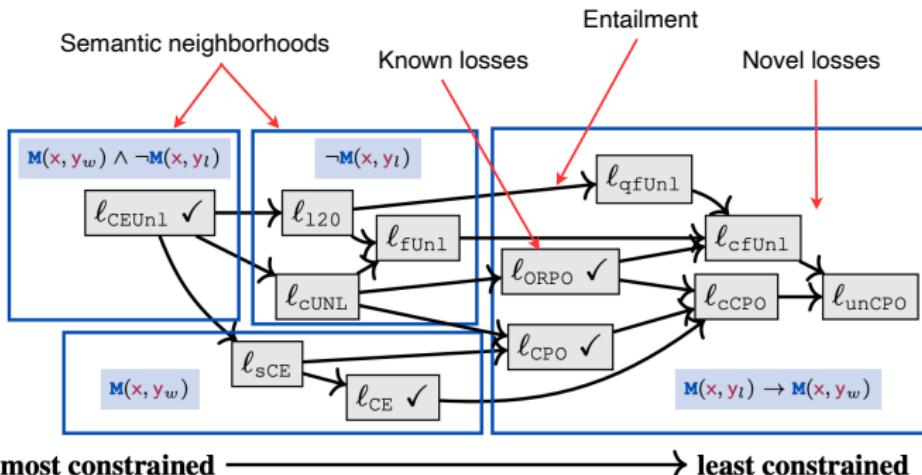
What properties do real losses have?

$M(x, y_w)$	$M(x, y_I)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	Implies($M(x, y_I), M(x, y_w)$)
T	F	✓	✓		
F	T	X	X		
F	F			✓ ✓ X ✓	Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Mapping out these loss spaces semantically

$$\begin{aligned}
 \underbrace{\ell_x}_{\text{column}} &:= -\log \sigma \left(\log \frac{\sum \checkmark_w}{\sum X_w} \right) \\
 &= -\log \sigma \left(\log \underbrace{\frac{\pi_\theta(y_I | x)\pi_\theta(y_w | x) + (1 - \pi_\theta(y_I | x))}{\pi_\theta(y_I | x)(1 - \pi_\theta(y_w | x))}}_{\text{novel loss without constraints, } P_\theta(P|T)} \right)
 \end{aligned}$$

The no reference loss landscape

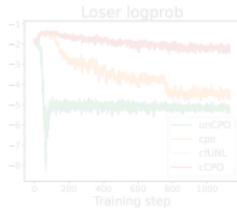


- **Loss lattice:** semantic structure of space, ordering.

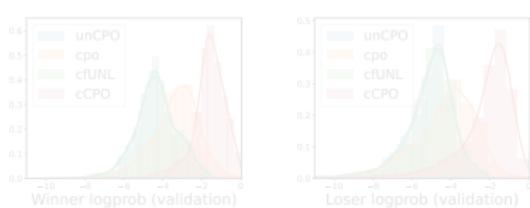
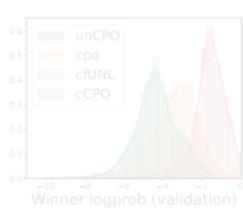
The no reference loss landscape



most constrained → least constrained

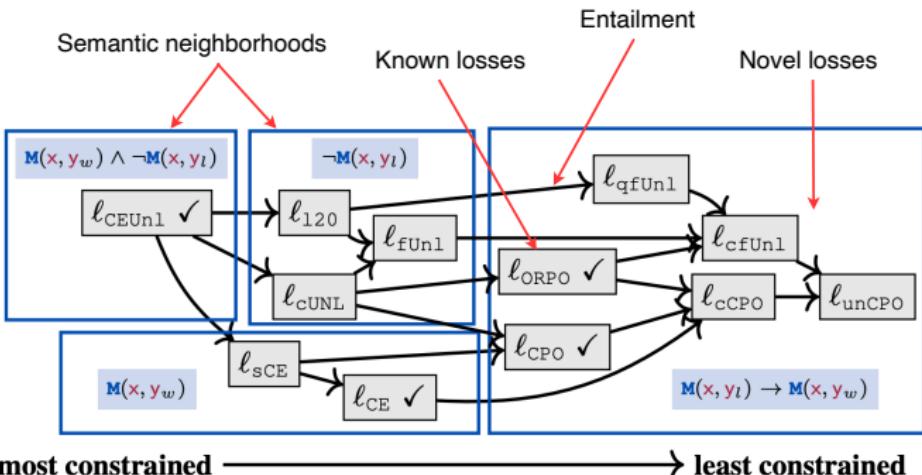


Training dynamics



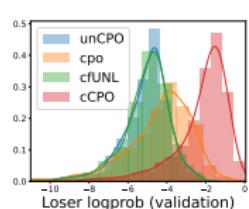
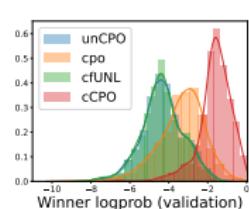
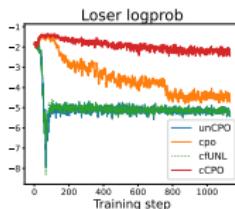
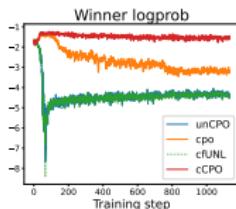
Inference

The no reference loss landscape



most constrained

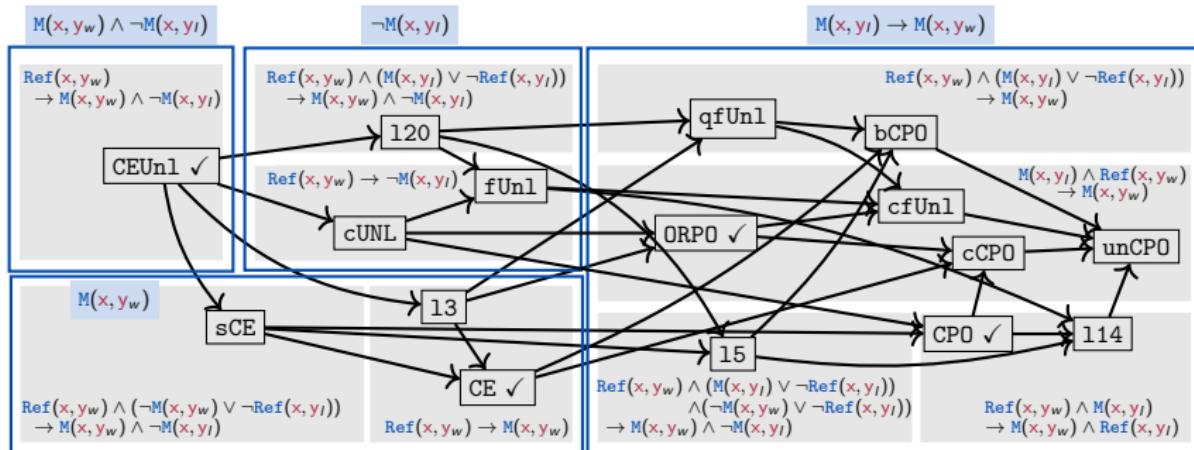
→ least constrained



Training dynamics

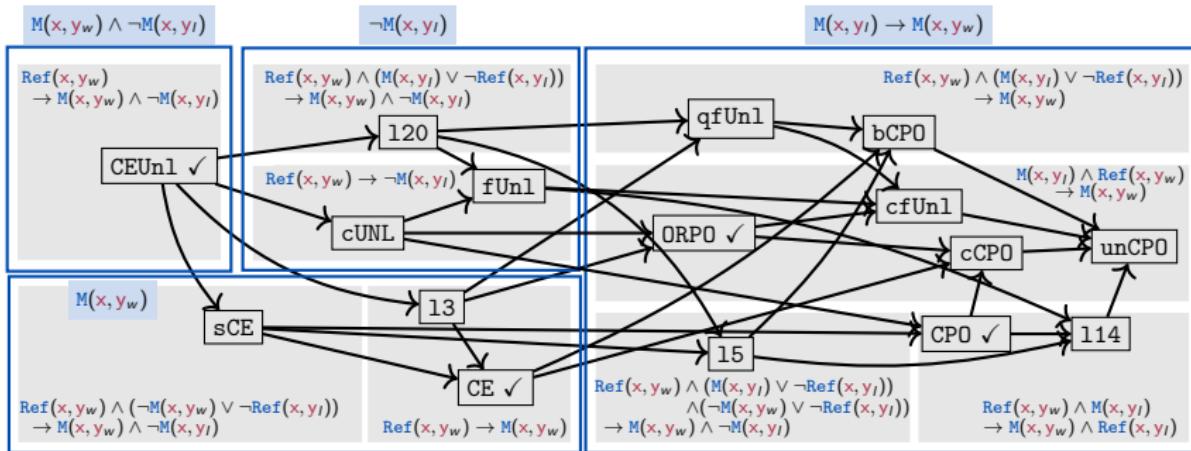
Inference

The full landscape, reference approaches



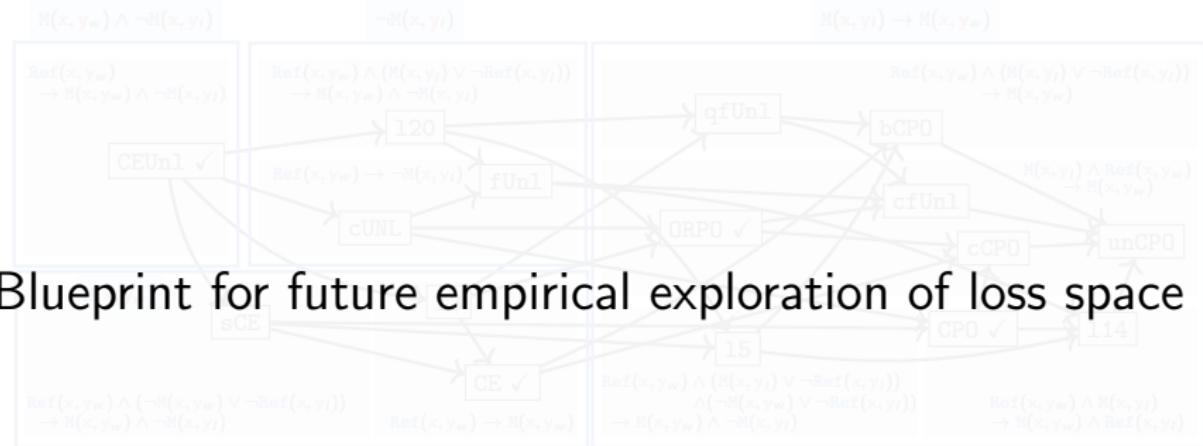
- ▶ The semantics of DPO-style reference losses can be straightforwardly computed from no reference approaches.

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,yl), Ref(x,yw)),  
    And(M(x,yw), Ref(x,yl)))  
)
```

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, yI), Ref(x, yw)),  
    And(M(x, yw), Ref(x, yI)))  
)
```

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

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

negating left side of implication (i.e., making $M(x, y_I)$ and $Ref(x, y_w)$ 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., Srikanth, 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*.