

# Quiz

- Go to **Quizzes > Quiz 10-27** on Canvas.
  - Session 17 Transformers part 1: J+M 7-7.2
  - Session 18 Transformers part 2: J+M 7.4-7.5, 7.9
- You have until **1:10pm** to complete it
- Allowed resources
  - Textbook
  - Your notes (on a computer or on paper)
  - Course slides and website
- Resources not allowed
  - Generative AI
  - Internet searches

# CS 1671 / CS 2071

## Human Language Technologies

Session 19: Post-training LLMs

---

Michael Miller Yoder

October 27, 2026

# Assessments: project deliverable 2

- [Project deliverable 2](#) is due **this Thu Oct 29**
- Part 1: Basic data analysis (if any updates are required from the proposal)
- Part 2: Result from baseline approach
  - Ideally performance metric result from the baseline system you proposed
- Part 3: Advanced system (LLM) proposal
  - How might you use an LLM programmatically to attempt your task?
  - Zero-shot and at least 1 more advanced approach

# Assessments: homework and exam

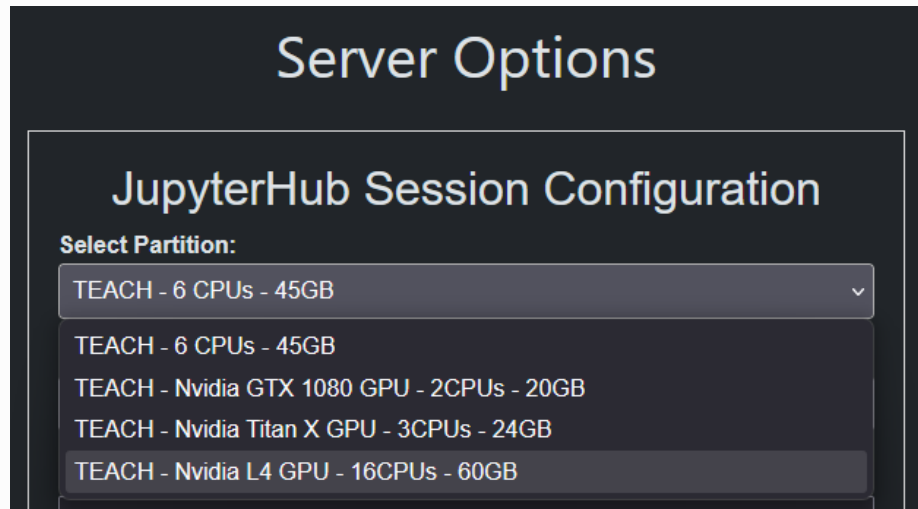
- Homework 3 will be released tomorrow
  - Instruction finetuning
- There will be an exam in class on **Tue Nov 10**
  - Covers everything in the class up to that point
  - In-class, paper exam
  - Conceptual
  - No programming
  - Class on Thu Nov 5 will be a review session for the exam

# Overview: Post-training LLMs

- Adapting LLMs to your use case
- Post-training and model alignment for LLMs
- Instruction tuning
- Preference-based learning
  - RLHF and DPO
- Coding activity: continued pretraining (finetuning) of GPT-2

# Load notebook: finetune GPT-2 on Shakespeare

1. Click this [nbgitpuller link](#) (also available on course website)
2. **Important difference from normal:**  
Start a server with 'TEACH – Nvidia L4 GPU – 16 CPUs – 60GB' server
3. Load custom environment at  
`/ix1/cs1671-2026f/class-env/.pixi/envs/default`
4. Open  
`session19_gpt2_shakespeare.ipynb`
5. Run the Setup cells



# Two ways of adapting an LLM to your use case

1. Finetune the parameters of the model with additional data
  - See parameter-efficient finetuning for finetuning very large models

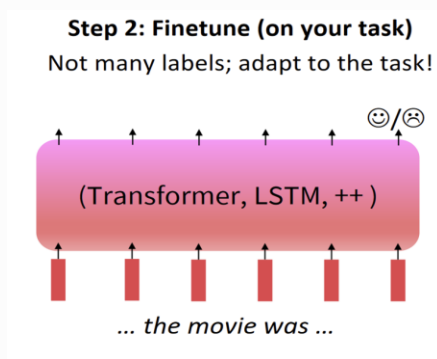
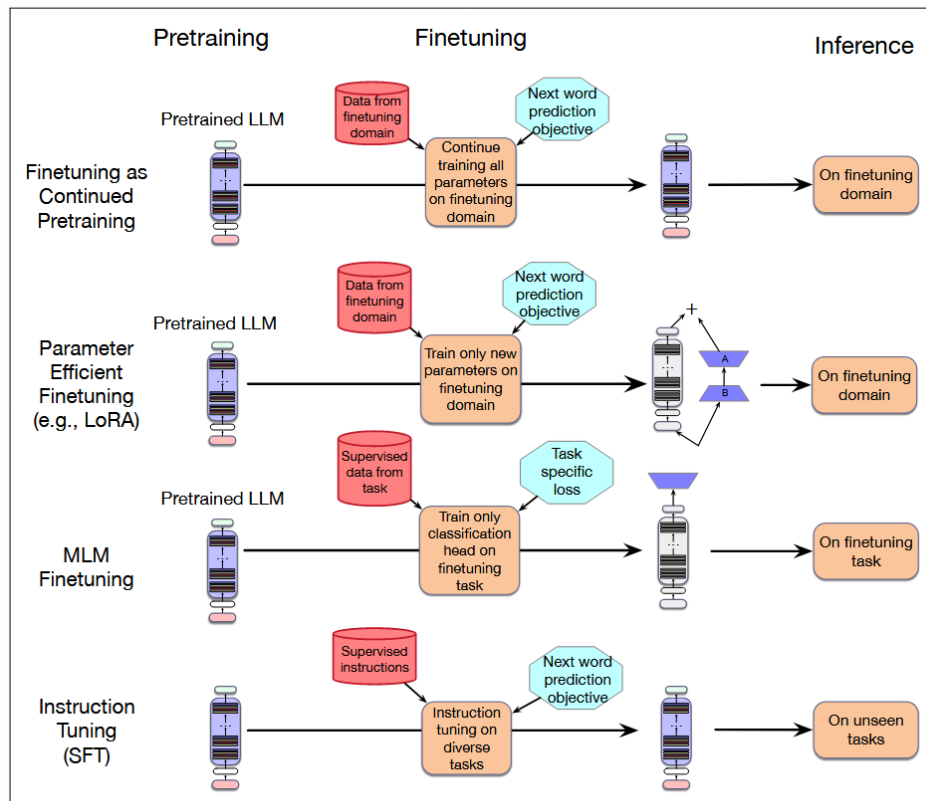


**Figure 9.1** Instruction tuning compared to the other kinds of finetuning.

# Two ways of adapting an LLM to your use case

2. Add more information in your prompt to the LLM, such as demonstrations (in-context learning)
  - In-context learning (few-shot prompting): give a few examples of output you want from the LLM
  - Chain-of-thought prompting: guide the LLM through a series of prompts that include intermediate reasoning

# Post-training and model alignment for LLMs: language modeling is not enough

---

# Language modeling $\neq$ assisting users

**PROMPT** *Explain the moon landing to a 6 year old in a few sentences.*

**COMPLETION** GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

- Language models are not aligned with user intent [Ouyang et al. 2022]
- Post-training LLMs to the rescue!

# Language model as multitask assistants?

- How do we get from
  - University of Pittsburgh is located in \_\_\_\_, Pennsylvania. [World knowledge]
  - I put \_\_ fork down on the table. [syntax]
- To...

# Language model as world models?

Language models may do rudimentary modeling of **agents**, **beliefs**, and **actions**.

*Pat watches a demonstration of a bowling ball and a leaf being dropped at the same time in a vacuum chamber. Pat, who is a physicist, predicts that the bowling ball and the leaf will fall at the same rate.*

Changing the last sentence of the prompt, we get:

*... Pat, who has never seen this demonstration before, predicts that the bowling ball will fall to the ground first. This is incorrect. In a vacuum chamber, there is no air*

# Language model as world models?

Language models may do **mathematical** proofs if pretrained on math content.

**Input:**  
Let  $f(r) = \sum_{j=2}^{2008} \frac{1}{j^r} = \frac{1}{2^r} + \frac{1}{3^r} + \cdots + \frac{1}{2008^r}$ . Find  $\sum_{k=2}^{\infty} f(k)$ .

---

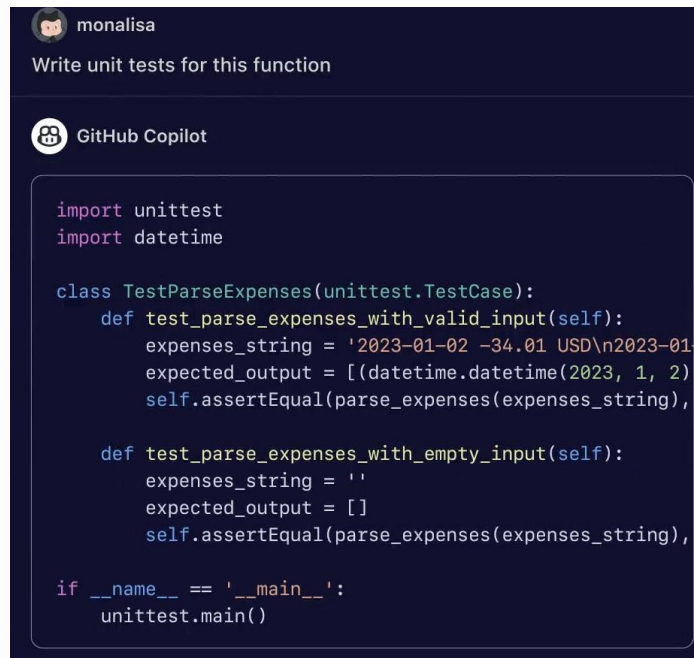
**LLEMMA 34B solution:**  
We have

$$\begin{aligned}\sum_{k=2}^{\infty} f(k) &= \sum_{k=2}^{\infty} \sum_{j=2}^{2008} \frac{1}{j^k} = \sum_{j=2}^{2008} \sum_{k=2}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \sum_{k=0}^{\infty} \frac{1}{j^k} = \sum_{j=2}^{2008} \frac{1}{j^2} \frac{1}{1 - \frac{1}{j}} \\ &= \sum_{j=2}^{2008} \frac{1}{j(j-1)} = \sum_{j=2}^{2008} \left( \frac{1}{j-1} - \frac{1}{j} \right) \\ &= \left( \frac{1}{1} - \frac{1}{2} \right) + \left( \frac{1}{2} - \frac{1}{3} \right) + \cdots + \left( \frac{1}{2007} - \frac{1}{2008} \right) \\ &= 1 - \frac{1}{2008} \\ &= \boxed{\frac{2007}{2008}}.\end{aligned}$$

Final Answer: The final answer is  $\frac{2007}{2008}$ .

# Language model as world models?

Language models can generate working code.



The screenshot shows a chat interface with a dark background. At the top, a user icon (a cat) is next to the name 'monalisa'. Below the name, the text 'Write unit tests for this function' is displayed. In the center, the GitHub Copilot logo (two stylized figures) is next to the text 'GitHub Copilot'. Below this, a code block contains the following Python code:

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 12.34 USD'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'),
                           (datetime.datetime(2023, 1, 3), 12.34, 'USD')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

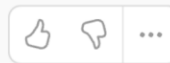
if __name__ == '__main__':
    unittest.main()
```

# Language model as multitask assistants?

**I need to throw a dinner party for 6 people who are vegetarian. Can you suggest a 3-course menu with a chocolate dessert?**

[[Microsoft Bing](#)]

(Also see OpenAI's ChatGPT, Google's Bard, Anthropic's Claude)



Sure, I can suggest a 3-course menu with a chocolate dessert for your vegetarian dinner party. Here are some possible options:

- Starter: **Wild Mushroom Tartlets with Onion Sauce** <sup>1</sup> or **Vegan Popcorn Tofu Nuggets** <sup>2</sup>
- Main: **Vegan Butternut Squash Mac** <sup>2</sup> or **Vegetarian Three-Bean Chili** <sup>3</sup>
- Dessert: **Chocolate Lava Cake** <sup>4</sup> or **Chocolate Pasta with Chocolate Hazelnut Cream Sauce, White Chocolate Shavings, and Fresh Berries** <sup>4</sup>

I hope this helps you plan your party. Enjoy! 😊

Learn more: [1. booths.co.uk](#) [+10 more](#)

# Post-training (model alignment)

Two techniques to align LLMs with human preferences (what we want them to do):

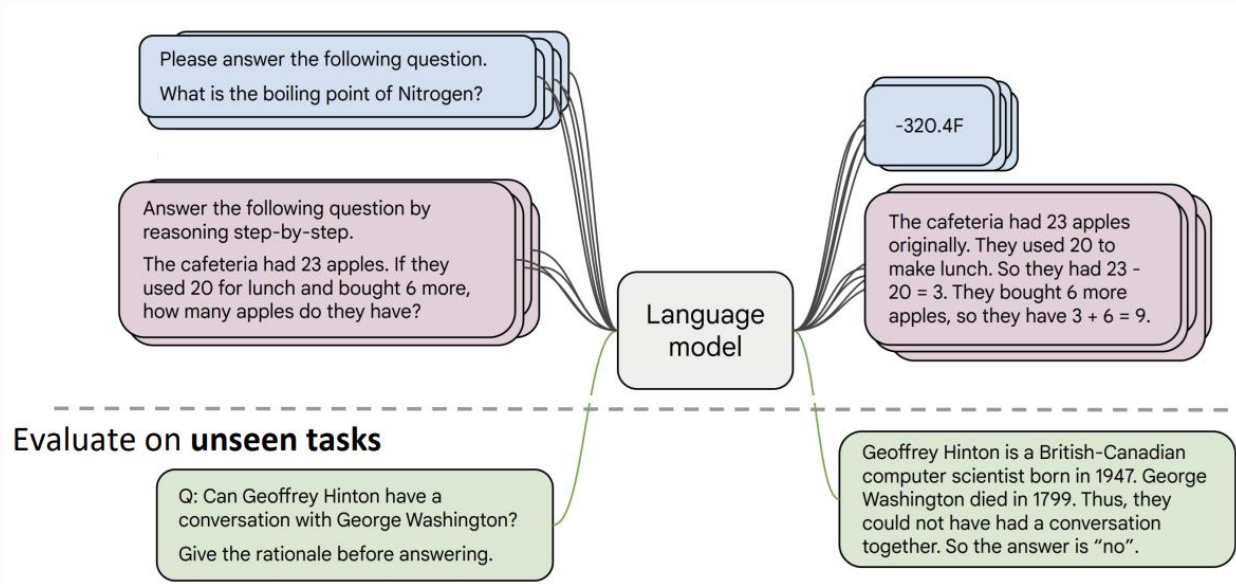
1. Instruction tuning
  - Models are finetuned on a corpus of instructions/questions and desired responses
2. Preference alignment (RLHF)
  - Separate model is trained to decide how much a candidate response aligns with human preferences
  - This reward model is used to finetune the base model

# Instruction tuning

---

# Instruction tuning

- AKA instruction finetuning, supervised finetuning, SFT
- Collect examples of (instruction, output) pairs across many tasks and finetune an LM
- Still just LM objective (predict the next word)



# Limitations of instruction finetuning

- Expensive to collect ground-truth data for tasks
  - Though you can include existing datasets of tasks like question answering
  - And LLMs are now commonly used to generate instruction tuning datasets
- Tasks like open-ended creative generation have no right answer.
  - Write me a story about a dog and her pet grasshopper.
- Language modeling penalizes all token-level mistakes equally, but some errors are worse than others
- Even with instruction finetuning, there is a mismatch between the LM objective and the objective of “satisfy human preferences”!
- Can we **explicitly attempt to satisfy human preferences**?

# Preference-based learning

---

# Preference-based alignment

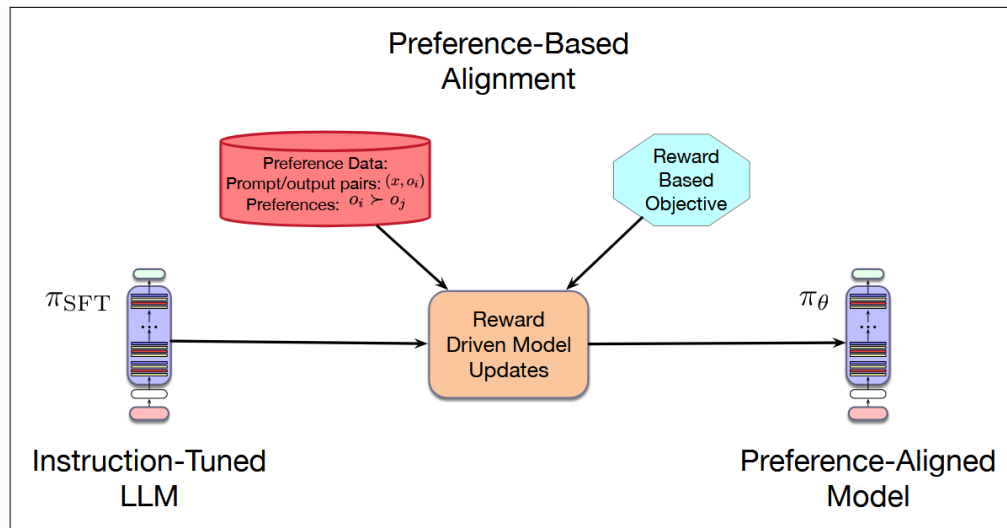


**Figure 9.8** Preference-based model alignment.

# How do we model human preferences? Preference data

- Ask annotators to rank different LLM responses by their preference
- Example: summarize an article

```
SAN FRANCISCO,  
California (CNN) --  
A magnitude 4.2  
earthquake shook the  
San Francisco  
...
```

```
An earthquake hit  
San Francisco.  
There was minor  
property damage,  
but no injuries.
```

>

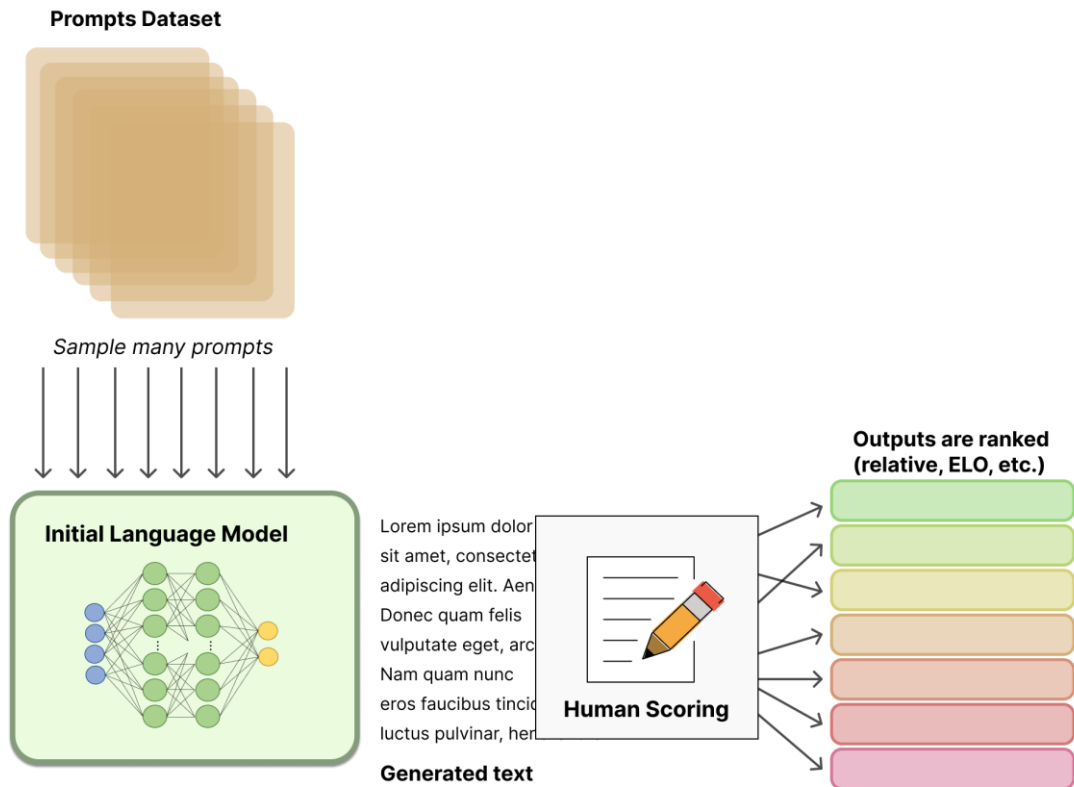
```
A 4.2 magnitude  
earthquake hit  
San Francisco,  
resulting in  
massive damage.
```

>

```
The Bay Area has  
good weather but is  
prone to  
earthquakes and  
wildfires.
```

# Preference data

- Prompts (can come from real users of OpenAI's LLMs, e.g.)
- LLM-generated responses to those prompts, ranked by human annotators



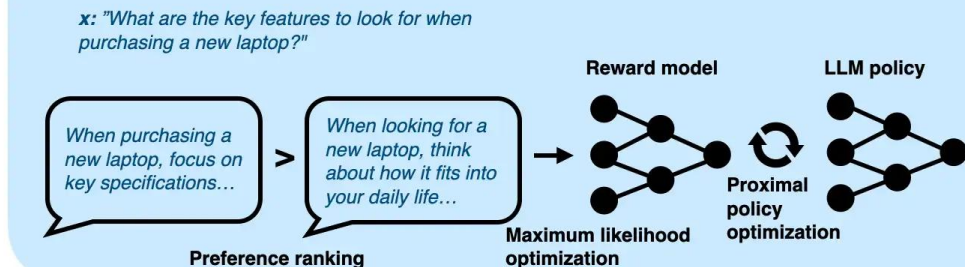
# Optimizing LLMs with preference data

- Preference-based learning often uses terms from reinforcement learning:
  - action: choosing tokens to generate
  - policy: an LLM that chooses the next word (a response)
  - reward: producing a response that matches human preferences
- We want to finetune the parameters of an LLM (predicting the next word) to maximize the expected “reward” for the entire output
- Try to not stray too far from the original (pretrained, instruction tuned) **reference language model**

# Algorithms for preference-based learning

- RLHF with PPO
  - Classic, used by LLMs like InstructGPT and original ChatGPT
  - Trains an explicit “reward model” that produces an estimate of human preference for any input text
- DPO
  - Doesn't use reinforcement learning
  - Newer (2023), more efficient
  - Doesn't train an explicit reward model

## Reinforcement Learning with Human Feedback (RLHF)



## Direct Preference Optimization (DPO)



# Finetuning LLMs with Direct Policy Optimization (DPO)

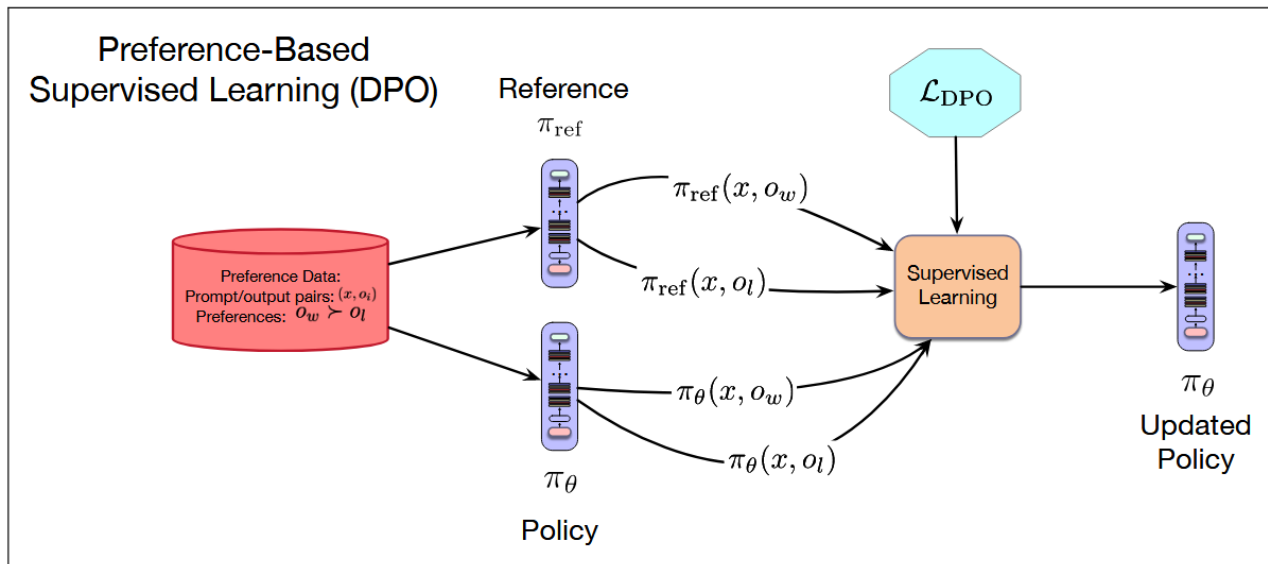


**Figure 9.9** Preference-based alignment with Direct Preference Optimization.

Calculates the LLM probabilities of the “winning” and “losing” responses from the reference model and the policy model that we are optimizing (finetuning)

# Loss function for Direct Policy Optimization (DPO)

Figure from Sebastian Raschka

The diagram illustrates the DPO loss function with various components and their meanings:

- LLM to optimize (called "policy")**: Points to  $\pi_\theta$  in the equation.
- Reference LLM**: Points to  $\pi_{ref}$  in the equation.
- DPO loss**: Points to the entire equation.
- Negative expected value**: Points to the negative sign and the expectation operator  $\mathbb{E}$ .
- Preference dataset**: Points to the dataset  $D$  in the expectation operator.
- $\beta$  is a hyperparameter**: Points to  $\beta$  in the equation.
- $y_w$  is the human preferred response**: Points to  $y_w$  in the numerator of the first log term.
- $x$  is a prompt from the dataset**: Points to  $x$  in the denominator of the first log term.
- $y_l$  is the human dispreferred response**: Points to  $y_l$  in the numerator of the second log term.
- Logistic sigmoid function**: Points to the  $\log \sigma$  part of the equation.
- Log probability of the human-preferred response that we want to maximize**: Points to the first log term,  $\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{ref}(y_w | x)}$ .
- Log probability of the human-dispreferred response that we want to minimize**: Points to the second log term,  $-\beta \log \frac{\pi_\theta(y_l | x)}{\pi_{ref}(y_l | x)}$ .

$$L_{DPO}(\pi_\theta; \pi_{ref}) = - \mathbb{E}_{(x, y_w, y_l) \sim D} \left[ \log \sigma \left( \underbrace{\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{ref}(y_w | x)}}_{\text{Log probability of the human-preferred response that we want to maximize}} - \underbrace{\beta \log \frac{\pi_\theta(y_l | x)}{\pi_{ref}(y_l | x)}}_{\text{Log probability of the human-dispreferred response that we want to minimize}} \right) \right]$$

- Loss will be high if the probability of the dispreferred response is larger than the preferred response
- Penalizes model for differing a lot from the reference model

# InstructGPT gets us closer to dialogue intent

**PROMPT** *Explain the moon landing to a 6 year old in a few sentences.*

**COMPLETION** GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

# InstructGPT can accomplish many tasks

**PROMPT** *Write a short poem about a wise frog.*

**COMPLETION** GPT-3

Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you have lost.

Write a short story in which a character has a bad dream.

InstructGPT

The frog is a symbol of wisdom  
He knows all the secrets of the world  
He is a master of disguise  
And a great teacher of life  
He is a symbol of transformation  
And the bringer of change  
He is the frog who has seen it all  
And knows the meaning of it all

# ChatGPT: Instruction Finetuning + RLHF for dialog agents

## ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge...

## Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

**(Instruction finetuning!)**

# ChatGPT: Instruction Finetuning + RLHF for dialog agents

## ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge...

## Methods

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

**(RLHF!)**

# Limitations of Reward Modeling

- Human preferences are unreliable!
- “Reward hacking” is a common problem in RL
- Chatbots are rewarded to produce responses that seem authoritative and helpful, regardless of truth
- This can result in making up facts + hallucinations

TECHNOLOGY

## Google shares drop \$100 billion after its new AI chatbot makes a mistake

February 9, 2023 · 10:15 AM ET

<https://www.npr.org/2023/02/09/1155650909/google-chatbot-error-bard-shares>

## Bing AI hallucinates the Super Bowl



<https://news.ycombinator.com/item?id=34776508>

<https://apnews.com/article/kansas-city-chiefs-philadelphia-eagles-technology-science-82bc20f207e3e4cf81abc6a5d9e6b23a>

# Conclusion

- Instruction tuning, finetuning of LLMs with prompt-response pairs, can help align language models with human preferences
- Large language models can be trained to provide more useful responses using preference-based learning
  - Types: Reinforcement learning from human feedback (RLHF), Direct policy optimization (DPO)
  - Finetune LLM parameters to prefer producing responses that humans preferred in preference training data
  - Don't change parameters too much from the initial (like instruction tuned) LLM

# Coding activity

---

# Notebook: finetune GPT-2 on Shakespeare

1. Click this [nbgitpuller link](#) (also available on course website)
2. **Important difference from normal:** Start a server with 'TEACH – Nvidia L4 GPU – 16 CPUs – 60GB' server
3. Load custom environment at `/ix1/cs1671-2026f/class-env/.pixi/envs/default`
4. Open `session19_gpt2_shakespeare.ipynb`

