Special Topic: Advanced NLP

Class 11: Instruction Finetuning for LLMs

Adam Faulkner

November 18, 2025

Why we need postraining to align with human preferences and instructions

- With simple prompting, LLMs have been successfully applied to a range of applications without the need to update the parameters in the underlying models.
- But there are limits to how much can be expected from a model whose sole training objective is to predict the next word from large amounts of pretraining text.

Why we need postraining to align with human preferences and instructions

- To see this, consider the following failed examples of following instructions from early work with GPT

Prompt: Explain the moon landing to a six year old in a few sentences.
Output: Explain the theory of gravity to a 6 year old.

Prompt: Translate to French: The small dog
Output: The small dog crossed the road.

- What went wrong? The LLM ignores the intent of the request and relies instead on its natural inclination to autoregressively generate continuations consistent with its context. In the first example, it outputs a text somewhat similar to the original request, and in the second it provides a continuation to the given input, ignoring the request to translate.
- LLMs are not sufficiently helpful: they need extra training to increase their abilities to follow textual instructions.

Instruction Tuning and Reinforcement Learning from Human Feedback

- To address this, most LLMs include two additional kinds of training for **model alignment**: methods designed to adjust LLMs to better align to human needs for models to be helpful and non-harmful.
  1. **Instruction tuning** (sometimes called **SFT** for **supervised finetuning**): Models are finetuned on a corpus of instructions and questions with their corresponding responses.
  2. **Reinforcement Learning from Human Feedback (RLHF)**: A separate model is trained to decide how much a candidate response aligns with human preferences. This model is then used to finetune the base model.
  3. **Direct Preference Optimization (DPO)**: A more efficient variant of RLHF.
- SFT, RLHF, and DPO are examples of **post-training**.

Finetuning an LLM to follow instructions

- Instruction tuning (IT) is a method for making an LLM better at following instructions.
- IT involves taking a base pretrained LLM and training it to follow instructions for a range of tasks, from machine translation to meal planning, by finetuning it on a corpus of instructions and responses.
- IT is a form of supervised learning where the training data consists of instructions
- Continue training the model on the instructions using the same language modeling objective used to train the original model: next-word prediction

IT is a form of supervised finetuning

- Even though it is trained to predict the next token this is **supervised finetuning (or SFT)** because, unlike in pretraining, each instruction or question in the instruction tuning data has a supervised objective: a correct answer to the question or a response to the instruction.
- Let's look at how IT differs from the kinds of finetuning we've already explored
- The goal of instruction tuning is not simply to learn a single task, but rather to learn to follow instructions in general, i.e., the LLM should be able to follow an instruction not seen in the training dataset, e.g, *Create a female teenage mutant ninja turtle character*

Four different forms of finetuning

- **Finetuning as continued pretraining**:Finetune as a way of adapting to a new domain by just continuing pretraining the LLM on data from a new domain (all parameters are updated). This is the form of finetuning we performed for our MLM-based finetuning.
- **Parameter-efficient finetuning** will be discussed next week: we create new (small) parameters and adapt them to the new domain
- **Task-based finetuning**: we adapt to a particular task by adding a new specialized classification head and updating its features via its own loss function (the parameters of the pretrained model may be frozen or might be slightly updated.)
- **Instruction tuning**, we take a dataset of instructions and their supervised responses and continue to train the language model on this data, based on the standard language model loss.

Instructions as training data

- **Instruction**: A natural language description of a task to be performed, combined with labeled task demonstrations. This can include minimal descriptions similar to the prompts weve already seen such as *Answer the following question*, *Create a resume based on the following information*, or *Summarize this report*
- Examples of instruction datasets:
  - **Aya**: 503 million instructions in 114 languages from 12 tasks including question answering, summarization, translation, etc.
  - **SuperNatural**: Instructions 12 million examples from 1600 tasks
  - **Flan 2022**: 15 million examples from 1836 tasks
  - **OPTIML**: 18 million examples from 2000 tasks

Instructions as training data

Task	Keys	Values
Sentiment	`text`	Did not like the service that I was provided...
	`label`	0
	`text`	It sounds like a great plot, the actors are first grade, and...
	`label`	0
NLI	`premise`	No weapons of mass destruction found in Iraq yet.
	`hypothesis`	Weapons of mass destruction found in Iraq.
	`label`	2
	`premise`	Jimmy Smith... played college football at University of Colorado.
	`hypothesis`	The University of Colorado has a college football team.
	`label`	0
Extractive Q/A	`context`	Beyoncé Giselle Knowles-Carter is an American singer...
	`question`	When did Beyonce start becoming popular?
	`answers`	{ `text`: ['in the late 1990s'], `answer_start`: 269 }

Figure 1: Examples of supervised training data for sentiment, natural language inference and Q/A tasks.

Instructions as training data

- **Definition:** This task involves creating answers to complex questions, from a given passage. Answering these questions, typically involve understanding multiple sentences. Make sure that your answer has the same type as the "answer type" mentioned in input. The provided "answer type" can be of any of the following types: "span", "date", "number". A "span" answer is a continuous phrase taken directly from the passage or question. You can directly copy-paste the text from the passage or the question for span type answers. If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words. A "number" type answer can include a digit specifying an actual value. For "date" type answers, use DD MM YYYY format e.g. 11 Jan 1992. If full date is not available in the passage you can write partial date such as 1992 or Jan 1992.
- **Emphasis:** If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words.
- **Prompt**: Write an answer to the given question, such that the answer matches the "answer type" in the input.
  Passage: { passage}
  Question: { question }

Figure 2: Example of a human crowdworker instruction from the **Natural Instructions** dataset for an extractive question answering task.

Instructing LLMs to provide safe responses

- Select questions from datasets of harmful questions (e.g., *How do I poison food?* or *How do I embezzle money?*)
- Then, expand this initial dataset by prompting an LLM to create multiple paraphrases of the questions (*Give me a list of ways to embezzle money*, etc.)
- Finally, generate a safe answers dataset by prompting an LLM to give safe answers to the questions (*I can't fulfill that request. Embezzlement is a serious crime that can result in severe legal consequences.*, etc.)

Limitation of IT

- Problem with IT:
  1. It's expensive to collect groundtruth data for tasks
  2. Language modeling penalizes all token-level mistakes equally, but some errors are worse than others.
  3. Tasks like open-ended creative generation have no right answer: *Write me a story about a dog and her pet grasshopper*
  4. Humans generate suboptimal answers
- Even with instruction finetuning, there a mismatch between the LM objective and the objective of "satisfy human preferences"!
- Can we explicitly attempt to satisfy human preferences?

Problem 1:It's expensive to collect groundtruth data for tasks

- Solution: Instead of directly asking humans for preferences, model their preferences as a separate (NLP) problem

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

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

$R(x, y_1) = 8.0$  $R(x, y_2) = 1.2$ 

- Specifically, train a **reward model** to predict human reward from an annotated dataset.

Problem 2: Human judgments are noisy and miscalibrated

- Solution: Instead of asking for direct ratings, ask for pairwise comparisons, which can be more reliable

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

$$R(x, y_3) = \overset{y_3}{4.1?} \quad 6.6? \quad 3.2?$$

Problem 2: Human judgments are noisy and miscalibrated

- Solution: Instead of asking for direct ratings, ask for pairwise comparisons, which can be more reliable

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.

y_1 1.2 y_3 y_2

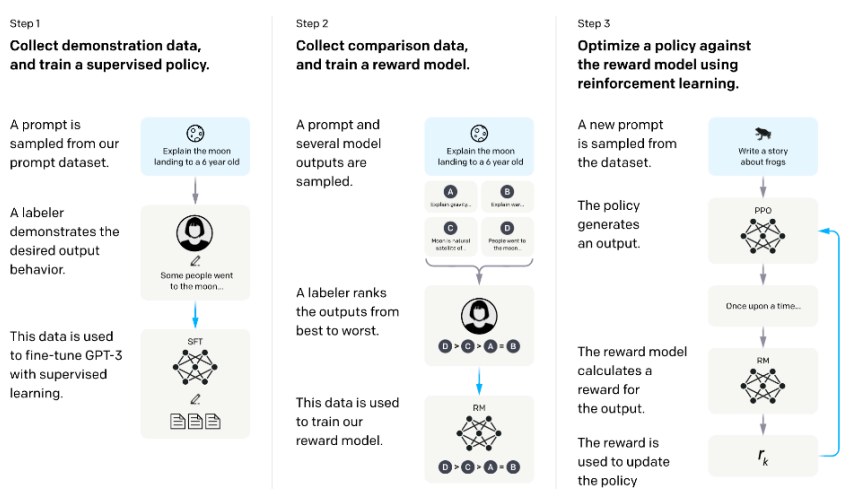
Introducing RLHF

- RLHF was introduced by OpenAI in the InstructGPTpaper (predecessor to ChatGPT)

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*
Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray
John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens
Amanda Askell† Peter Welinder Paul Christiano*†
Jan Leike* Ryan Lowe*
OpenAI

Introducing RLHF



Notation for RLHF: LLM

- **Input**: Context or prompt
- **Model:** π with parameters θ , used to generate text as output.
- **Preference Dataset:** For any given context, there exists a pair of outputs: one preferred and one not preferred. π with parameters θ , used to generate text as output. Preference Dataset: For any given context, there exists a pair of outputs: one preferred and one not preferred.

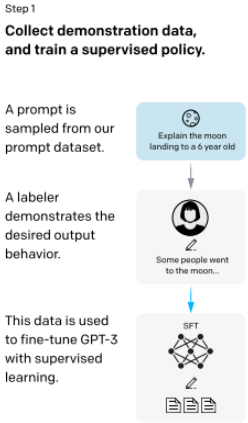
Notation for RLHF: Reinforcement Learning (RL)

- **RL:** a type of machine learning where an agent learns to make decisions by performing actions in an environment to achieve a goal. The agent receives feedback in the form of rewards or penalties, which guide its learning process.
- **Agent:** This is the learner or decision-maker. It's the program or model that is being trained.
- **Environment**: The world through which the agent moves, which provides specific states to the agent. The environment could be a real-world setting, a computer game, a simulated environment, etc. In RLHF, the environment provides a single episode preference, as given by the preference dataset.
- **Actions:** These are the choices made by the agent. The set of all possible actions is known as the action space.

Notation for RLHF: Reinforcement Learning (RL)

- **State:** A state is a concrete and immediate situation in which the agent finds itself; its a snapshot of the environment at a given time.
- **Reward:** After each action, the agent receives a reward (or penalty) from the environment. This reward is a numerical value that reflects the success or failure of the action.

Step 1 (Supervised Finetuning): Collect demonstration data and train a supervised policy



Step 1 (Supervised Finetuning): Collect demonstration data and train a supervised policy

- From InstructGPT paper:

Tasks collected from labelers:

- **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
- **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
- **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.

Step 1 (Supervised Finetuning): Collect demonstration data and train a supervised policy

- From InstructGPT paper:

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

COMPLETION

GPT-3

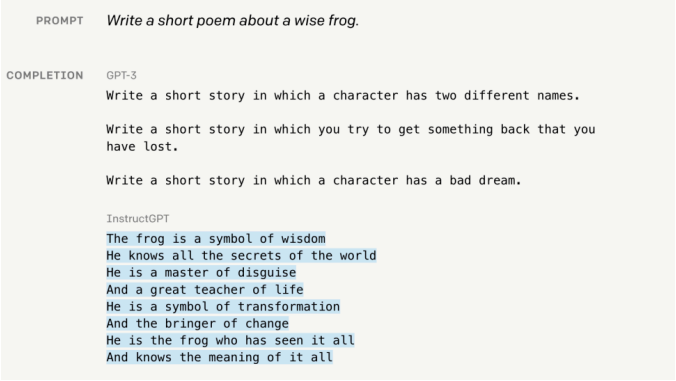
```
Explain the theory of gravity to a 6 year old.

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.
```

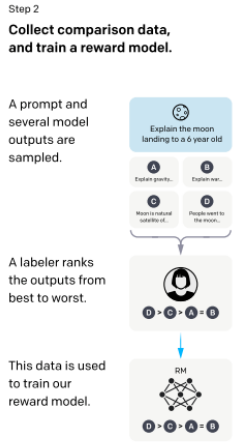
Step 1 (Supervised Finetuning): Collect demonstration data and train a supervised policy

- From InstructGPT paper:

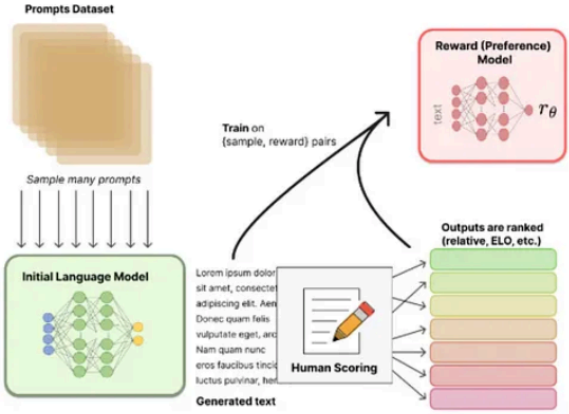


Step 2: Collect comparison data and train a reward model

- From InstructGPT paper:



Step 2: Collect comparison data and train a reward model



Step 2: Collect comparison data and train a reward model

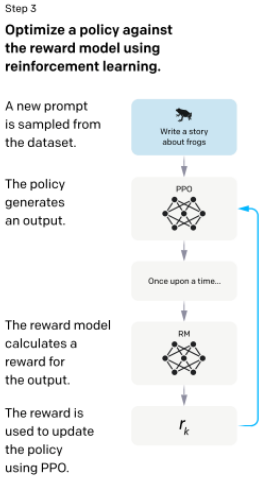
- In the second step of RLHF, the focus is on learning a reward model that will guide the reinforcement learning (RL) process in the subsequent step.
- To train this reward model, a dataset with preferences is essential.
- Initially, this dataset consists of human-written prompts, AI-generated responses, and corresponding human ratings.
- The loss function used is maximizing the likelihood that the preferred response has a higher probability than the non-preferred one. (This comparison probability is modeled by the classic Bradley-Terry model, which we won't be reviewing)

Step 2: Collect comparison data and train a reward model

- The goal is, given an LLM that takes in a sequence of text, return a scalar reward that numerically represents the human preference.
- The output being a scalar reward is crucial for existing RL algorithms being integrated seamlessly later in the RLHF process.

Step 3: Optimize a policy against the reward model using RL

- From InstructGPT paper:



Step 3: Optimize a policy against the reward model using RL

- Context: In the 2010s, a resurgence of interest in RL applied to deep learning, game-playing (AlphaGo)
- But the interest in applying RL to modern LMs is an even newer phenomenon
- LM application of RL:
  - Generate completions from a reward policy p_θ^{RL} for several tasks
  - Compute reward using $RM(x, y)$
  - Update$p_\theta^{RL}(y|x)$ to increase probability of high-reward completions

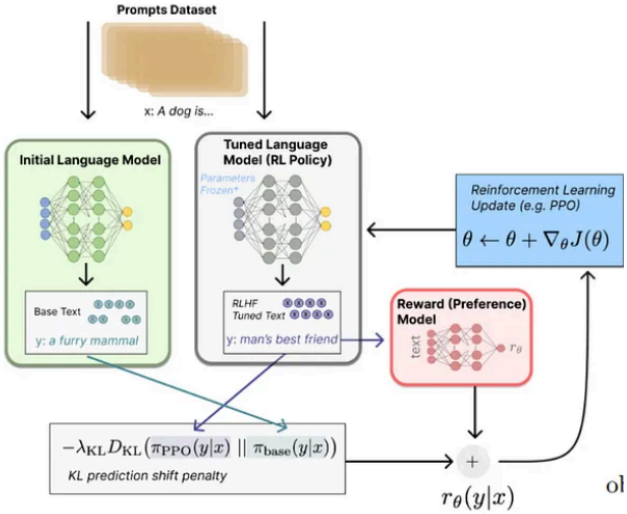
Step 3: Optimize a policy against the reward model using RL

- At this point in the RLHF system, we have an initial language model that can be used to generate text and a preference model that takes in any text and assigns it a score of how well humans perceive it.
- Next, we use RL to optimize the original language model with respect to the reward model.

Step 3: Optimize a policy against the reward model using RL

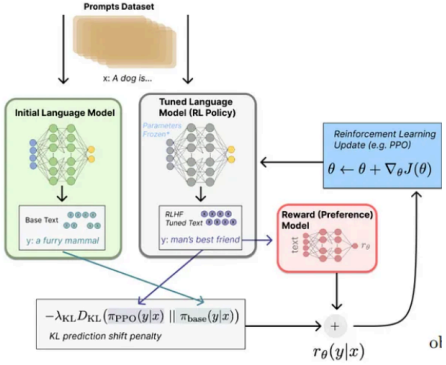
- Let's first formulate this fine-tuning task as a RL problem.
- **First,** the **policy** is a language model that takes in a prompt and returns a sequence of text (or just probability distributions over text).
- **Second,** the **action space** of this policy is all the tokens corresponding to the vocabulary of the language model (often on the order of 50k tokens).
- **Third,** the **observation space** is the distribution of possible input token sequences, which is also quite large given previous uses of RL.
- Finally, the **reward function** is a combination of the preference model and a constraint on policy shift.

Step 3: Optimize a policy against the reward model using RL



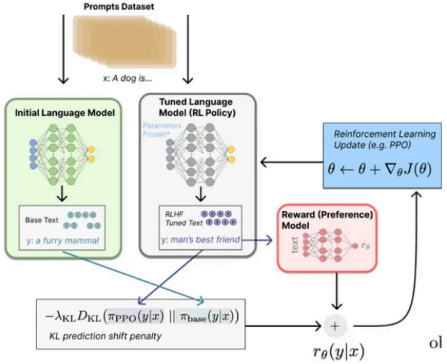
Step 3: Optimize a policy against the reward model using RL

- Given a prompt, x, from the dataset, the text y is generated by the current iteration of the fine-tuned policy. Concatenated with the original prompt, that text is passed to the preference model, which returns a scalar notion of "preferability", r_θ



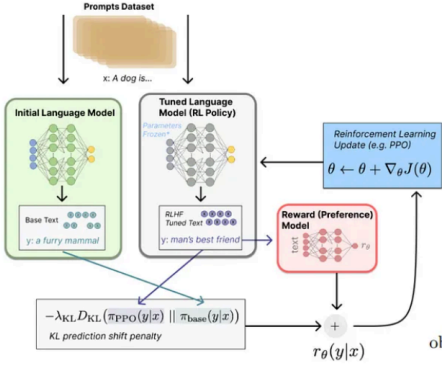
Step 3: Optimize a policy against the reward model using RL

- In addition, per-token probability distributions from the RL policy are compared to the ones from the initial model to compute a penalty on the difference between them: the Kullback-Leibler (KL) divergence between these sequences of distributions over tokens, r_{KL}



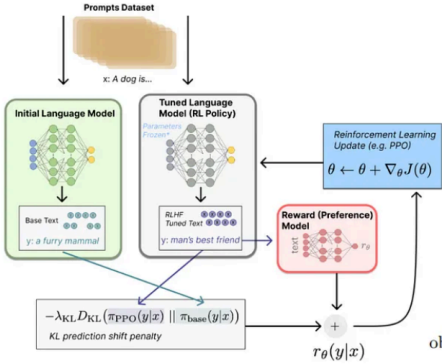
Step 3: Optimize a policy against the reward model using RL

- The KL divergence term penalizes the RL policy from moving substantially away from the initial pretrained model with each training batch—this is to ensure that the model outputs reasonably coherent text snippets.



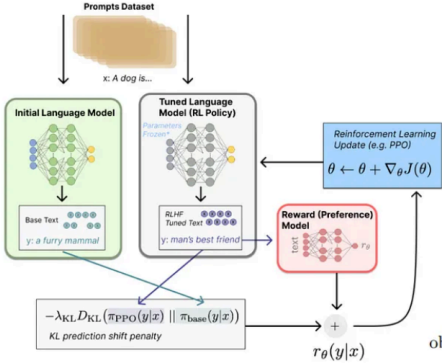
Step 3: Optimize a policy against the reward model using RL

- The final reward sent to the RL update rule is $r = r_\theta - \lambda_{rKL}$



Step 3: Optimize a policy against the reward model using RL

- Finally, the update rule is the parameter update from PPO that maximizes the reward metrics in the current batch of data



Limitations of RL + 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“)

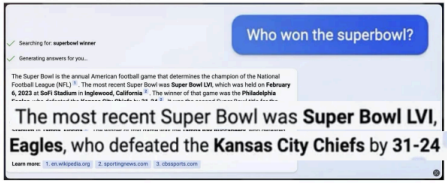
TECHNOLOGY

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

February 9, 2023 · 10:15 AM ET

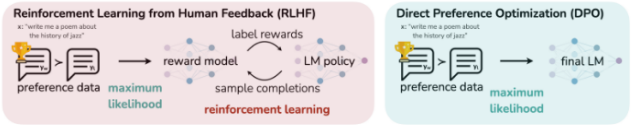
https://www.npr.org/2023/02/09/1155650909/google-chatbot--error-bard-shares

Bing AI hallucinates the Super Bowl



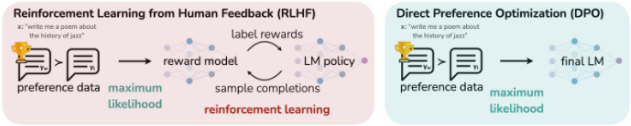
Introducing Direct Preference Optimization (DPO)

- In traditional RLHF, two separate stages are involved: one to build a reward model and another to fine-tune the language model via reinforcement learning.
- DPO, on the other hand, employs a single-stage policy training that directly solves a classification problem based on human preference data.



Introducing Direct Preference Optimization (DPO)

- Instead of collecting feedback to create a reward model and then fine-tuning the LM to adhere to it, DPO directly trains the LM to respond according to a dataset of preferred human responses.



Most LLMs are now post-trained using DPO rather than RLHF

- **Reinforcement learning from human feedback**
  - Train an explicit reward model on comparison data to predict a score for a given completion
  - Optimize the LM to maximize the predicted score (under KL-constraint)
  - Very effective when tuned well, computationally expensive and tricky to get right
- **Direct Preference Optimization**
  - Optimize LM parameters directly on preference data by solving a binary classification problem
  - Simple and effective, similar properties to RLHF, does not leverage online data

Next class: November 25

Assignment 4 Due: Midnight 11/25

Augmented LLMs

- ReAct: Synergizing Reasoning and Acting in Language Models
- Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks
- RAGAS: Automated Evaluation of Retrieval Augmented Generation
- Agents (HF tutorial)
- Exploring Large Language Model Based Intelligent Agents: Definitions, Methods, and Prospects