Post-training

Core idea:

to find where is that intelligence coming from?

World model

Main function:

Understand the current state and predict the future state without having to try every possible scenario in the real world

Concrete manifestation

LLM, video generation model (eg. sora)

what can we do with it?

Planning and Reasoning, Efficient leaning of samples,

Dealing with partial observability, understanding of causality

---

① ZS & FS

⚠ Zero-shot learning (ZSL)

Why did we develop it? (emerge)

As we pretrained on more and more data and more and more tasks, we start seething this phenomenon where they're able to do the task basically zero shot. Then we tend to let model learn more abstract and essential features

↓

ZSL emerged in GPT-2

Why is it good ?

It doesn't require the test categories to appear in the training set. Its goal is to enable the model to learn a general recognition ability, so that it can transfer knowledge to unknown categories

How can it do that ?

by Semantic representation

1. training: learn the correlation between "seeing" and describing

2. testing/reasoning: Search for the best match in semantic space

(2) Few-Shot learning

core idea: teach the model "how to recognize new things"

basic setting is N-way K-shot

N categories, K labeled examples

(2) Instruction Finetuning

Why do we need it?

LLM have a big problem is that the capability of the model doesn't align with the user's intention

① differ with pretraining

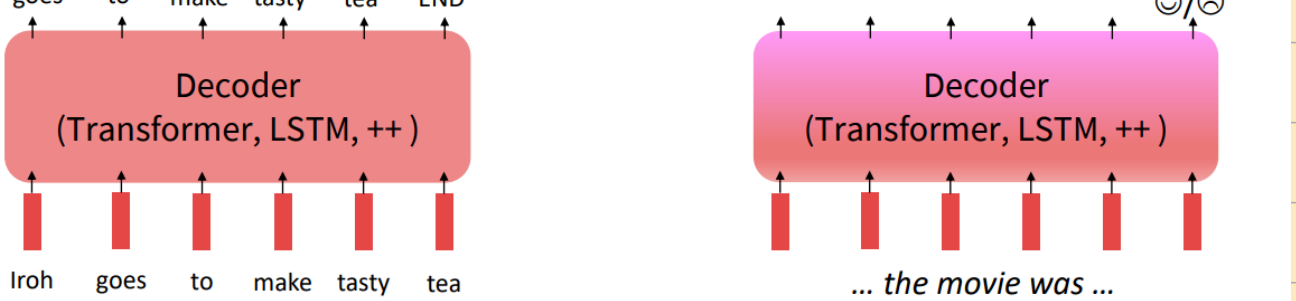
Scaling up finetuning

Pretraining can improve NLP applications by serving as parameter initialization.

Step 1: Pretrain (on language modeling)
Lots of text; learn general things!

Step 2: Finetune (on many tasks)
~~Not~~ many labels; adapt to the tasks!

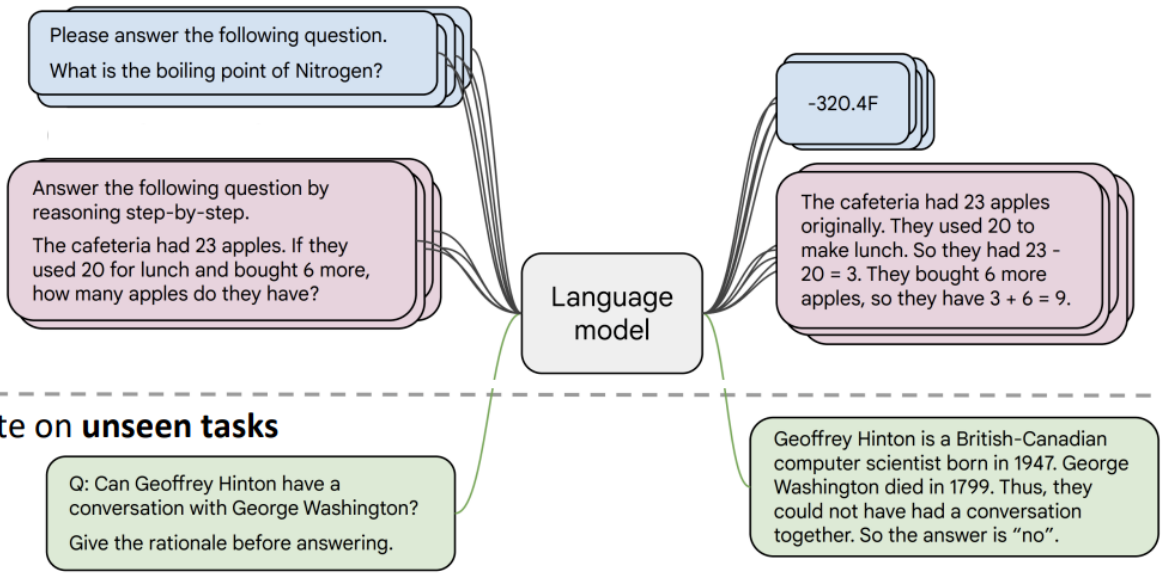
goes to make tasty tea END



② paradigm

Instruction finetuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



- Evaluate on **unseen tasks**

③ how to evaluate such a model?

MMLU

example

Massive Multitask Language Understanding (MMLU)
[Hendrycks et al., 2021]

New benchmarks for measuring LM performance on 57 diverse *knowledge intensive* tasks

Astronomy

What is true for a type-Ia supernova?
A. This type occurs in binary systems.
B. This type occurs in young galaxies.
C. This type produces gamma-ray bursts.
D. This type produces high amounts of X-rays.
Answer: A

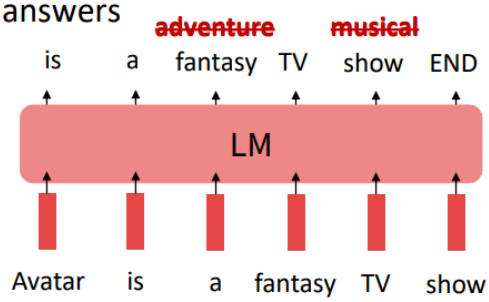
High School Biology

In a population of giraffes, an environmental change occurs that favors individuals that are tallest. As a result, more of the taller individuals are able to obtain nutrients and survive to pass along their genetic information. This is an example of
A. directional selection.
B. stabilizing selection.
C. sexual selection.
D. disruptive selection
Answer: A

④ Limitations

Limitations of instruction finetuning?

- One limitation of instruction finetuning is obvious: it's **expensive** to collect ground-truth data for tasks. Can you think of other subtler limitations?
- **Problem 1:** tasks like open-ended creative generation have no right answer.
  - *Write me a story about a dog and her pet grasshopper.*
- **Problem 2:** language modeling penalizes all token-level mistakes equally, but some errors are worse than others.
- **Problem 3:** 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?**



Q. A. Does pretraining also have problem 2?
↳ Yes

(3) Optimizing for human preferences

Optimizing for human preferences

- Let's say we were training a language model on some task (e.g. summarization).
- For an instruction x and a LM sample y, imagine we had a way to obtain a *human reward* of that summary: $R(x, y) \in \mathbb{R}$, higher is better.

x:
```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.
```

y_1:
```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```
$R(x, y_1) = 8.0$

y_2:
```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```
$R(x, y_2) = 1.2$

- Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{y} \sim p_\theta(y \mid x)}[R(x, \hat{y})]$$

$R(x, y)$ is a reward function

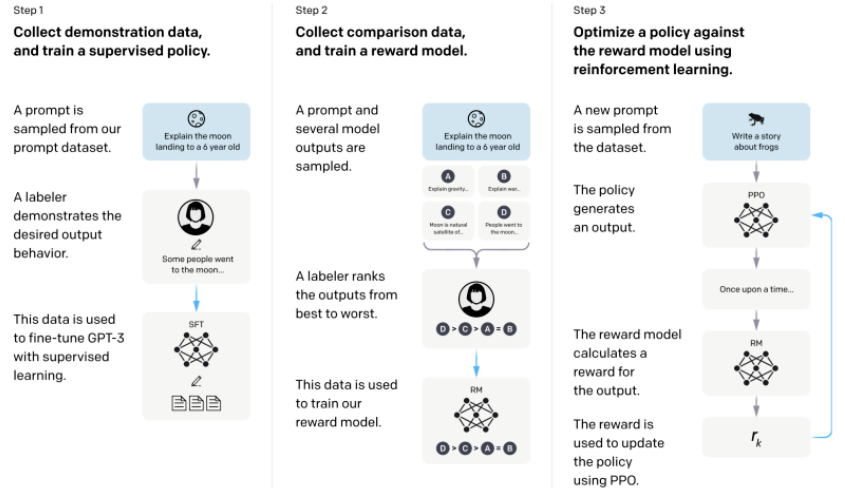
input x (prompt)
↓
model $p_\theta(y|x)$ generate the answer $\hat{y}$
↓
reward model $R(x, \hat{y})$
↓
output reward score

↓

adjust p_θ to increase the probability of high-reward responses

(2) RLHF (Reinforcemet Learning from Human Feed back)

High-level instantiation: 'RLHF' pipeline



- First step: instruction tuning!
- Second + third steps: maximize reward (but how??)

Teach model to understand the tasks

↓

Define the crieria for "what constitutes a good answer"

↓

Let model optimizes itself

2nd 3rd

How do we get the rewards?

- **Problem 1:** human-in-the-loop is expensive!
  - **Solution:** instead of directly asking humans for preferences, **model their preferences** as a separate (NLP) problem! [Knox and Stone, 2009]

```
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.
```

Train a $RM_\phi(x, y)$ to predict human reward from an annotated dataset, then optimize for RM_ϕ instead.

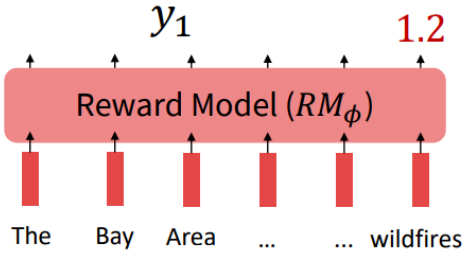
$R(x, y_1) = 8.0$ $R(x, y_2) = 1.2$

↳ this is a simple machine learning regression style problem

How do we model human preferences?

- **Problem 2:** human judgments are noisy and miscalibrated!
- **Solution:** instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable [Phelps et al., 2015; Clark et al., 2018]

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_3 y_2

Bradley-Terry [1952] paired comparison model

$$J_{RM}(\phi) = -\mathbb{E}_{(x, y^w, y^l)\sim D}\left[\log \sigma(RM_\phi(x, y^w) - RM_\phi(x, y^l))\right]$$

"winning" sample — "losing" sample — y^w should score higher than y^l

$\sigma(x)$ sigmoid function

$$\log \sigma(RM_\phi(x, y^w) - RM_\phi(x, y^l))$$

↳ this is a pairwise ranking loss

≈ the logarithm of the probability that good sample score is higher than a bad one

So, we want to maximize it or minimize its negative value

↓

RLHF: Optimizing the learned reward model

- We want to optimize:

$$\mathbb{E}_{\hat{y}\sim p_\theta^{RL}(\hat{y}|x)}\left[RM_\phi(x, \hat{y})\right]$$

- Do you see any problems?
  - Learned rewards are imperfect; this quantity can be imperfectly optimized
- Add a penalty for drifting too for from the initialization:

$$\mathbb{E}_{\hat{y}\sim p_\theta^{RL}(\hat{y}|x)}\left[RM_\phi(x, \hat{y}) - \beta \log\left(\frac{p_\theta^{RL}(\hat{y} \mid x)}{p^{PT}(\hat{y} \mid x)}\right)\right]$$

Pay a price when $p_\theta^{RL}(\hat{y} \mid x) > p^{PT}(\hat{y} \mid x)$

This penalty which prevents us from diverging too far from the pretrained model. In expectation, it is known as the **Kullback-Leibler (KL)** divergence between $p_\theta^{RL}(\hat{y} \mid x)$ and $p^{PT}(\hat{y} \mid x)$.

→ why should we do that?

L we know the initalization of the model is a decent LM

p^{RT} : is the reference model which typically the SFT (supervised Fine-Tuning) model obtained in the 1st step

p^{RL} : the current policy model being optimized by RL (reinforcement learning)

$\frac{p^{RL}}{p^{RT}}$: The "preference" level of the current model for a certain answer relative to the reference model

problems

1. reward hack

   RM maybe prefer some long text, so model will abuse such words

2. Pattern collapse and language degradation

   To maximize reward, model may converge to a very small output space, generate some "safe" sentences which can get high score.

3. catatrophic forgetting

   model may forget some valuable language knowledges or skills which it learned from pretraining & instruction finetuning steps

Simplification (DPO)

Can we simplify RLHF? Towards **Direct Preference Optimization**

- Current pipeline is as follows:
  - Train a reward model $RM_\phi(x, y)$ to produce scalar rewards for LM outputs, trained on a **dataset of human comparisons**
  - Optimize pretrained (possibly instruction-finetuned) LM $p^{PT}(y \mid x)$ to produce the final RLHF LM $p_\theta^{RL}(\hat{y} \mid x)$
- What if there was a way to write $RM_\phi(x, y)$ in terms of $p_\theta^{RL}(\hat{y} \mid x)$?
  - Derive $RM_\theta(x, y)$ in terms of $p_\theta^{RL}(\hat{y} \mid x)$
  - Optimizing parameters θ by fitting $RM_\theta(x, y)$ to the preference data instead of $RM_\phi(x, y)$
- How is this possible? The only external information to the optimization comes from the preference labels

This means: since both the reward model and the optimized strategy model are derived from the same set of human preference data, there should be a direct mathematical relationship between term

⇓

direct preference optimization

core idea: Write a reward model in terms of LM itself

don't need RL at all

Direct Preference Optimization (DPO)

- Recall, we want to maximize the following objective:

$$\mathbb{E}_{\hat{y}\sim p_\theta^{RL}(\hat{y}|x)}\left[RM(x,\hat{y}) - \beta \log\left(\frac{p_\theta^{RL}(\hat{y}\mid x)}{p^{PT}(\hat{y}\mid x)}\right)\right]$$

闭式解

- There is a closed form solution to this:

$$p^*(\hat{y}\mid x) = \frac{1}{Z(x)} p^{PT}(\hat{y}\mid x)\exp\left(\frac{1}{\beta}RM(x,\hat{y})\right)$$

- Rearrange the terms:

* 2(x) -> / p^{PT}
log = log

$$RM(x,\hat{y}) = \beta \log\frac{p^*(\hat{y}\mid x)}{p^{PT}(\hat{y}\mid x)} + \beta \log Z(x)$$

- This holds true for arbitrary LMs

$$RM_\theta(x,\hat{y}) = \beta \log\frac{p_\theta^{RL}(\hat{y}\mid x)}{p^{PT}(\hat{y}\mid x)} + \beta \log Z(x)$$

Z(x) is normalization function

this formula can be intuitively understood as: the optimal strategy is to assign exponentially higher probability to those high reward outputs based on the original strategy

Direct Preference Optimization (DPO)

- Recall, how we fit the reward model $RM_\phi(x,y)$:

$$J_{RM}(\phi) = -\mathbb{E}_{(x,y^w,y^l)\sim D}\left[\log\sigma(RM_\phi(x,y^w) - RM_\phi(x,y^l))\right]$$

- Notice that we only need the **difference** between the rewards for y^w and y^l. Simplify for $RM_\theta(x,y)$:

$$RM_\theta(x,y^w) - RM_\theta(x,y^l) = \beta \log\frac{p_\theta^{RL}(y^w\mid x)}{p^{PT}(y^w\mid x)} - \beta \log\frac{p_\theta^{RL}(y^l\mid x)}{p^{PT}(y^l\mid x)}$$

- The final DPO loss function is:

$$J_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y^w,y^l)\sim D}\left[\log\sigma\left(RM_\theta(x,y^w) - RM_\theta(x,y^l)\right)\right]$$

We have a *simple classification loss* function that connects **preference data** to **language model parameters** directly!

we find $Z(x)$, the partition function is disappered and this partition function is hard to compute, but we don't need to compute now.

we don't need to explicitly train a reward function, we can directly use human comparision data and optimize model parameters through standard supervised learning (gradient descent)

Summary (DPO and RLHF)

- We want to optimize for human preferences
  - Instead of humans writing the answers or giving uncalibrated scores, we get humans to rank different LM generated answers
- 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

InstructGPT: scaling up RLHF to tens of thousands of tasks

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.

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!)

GPT-3 => InstructGPT => chatGPT

Lecture Plan: From Language Models to Assistants

1. **Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning**
   - \+ No finetuning needed, prompt engineering (e.g. CoT) can improve performance
   - – Limits to what you can fit in context
   - – Complex tasks will probably need gradient steps
2. **Instruction finetuning**
   - \+ Simple and straightforward, generalize to unseen tasks
   - – Collecting demonstrations for so many tasks is expensive
   - – Mismatch between LM objective and human preferences
3. **Reinforcement Learning from Human Feedback (RLHF)**
   - \+ Directly model preferences (cf. language modeling), generalize beyond labeled data
   - – RL is very tricky to get right
   - – Human preferences are fallible; *models* of human preferences even more so
4. **What's next?**

(1) Transformer ← model architecture

↓

Pretraining ← learn languages rules

↓

Instruction Finetuning ← learn following instruction

↓

RLHF/DPO ← learn satisfying human

↓

Complete aligning ← ChatGPT, Claude, Gemini...

Be aware, Instruction Finetuning is a specific form of Supervised finetuning (SFT)