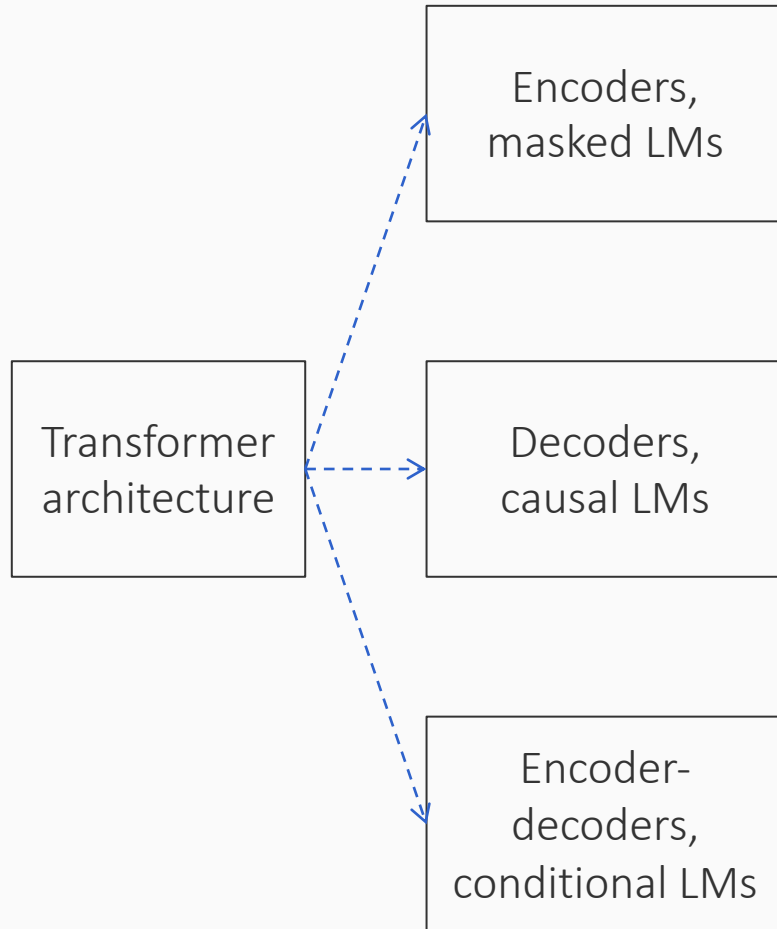
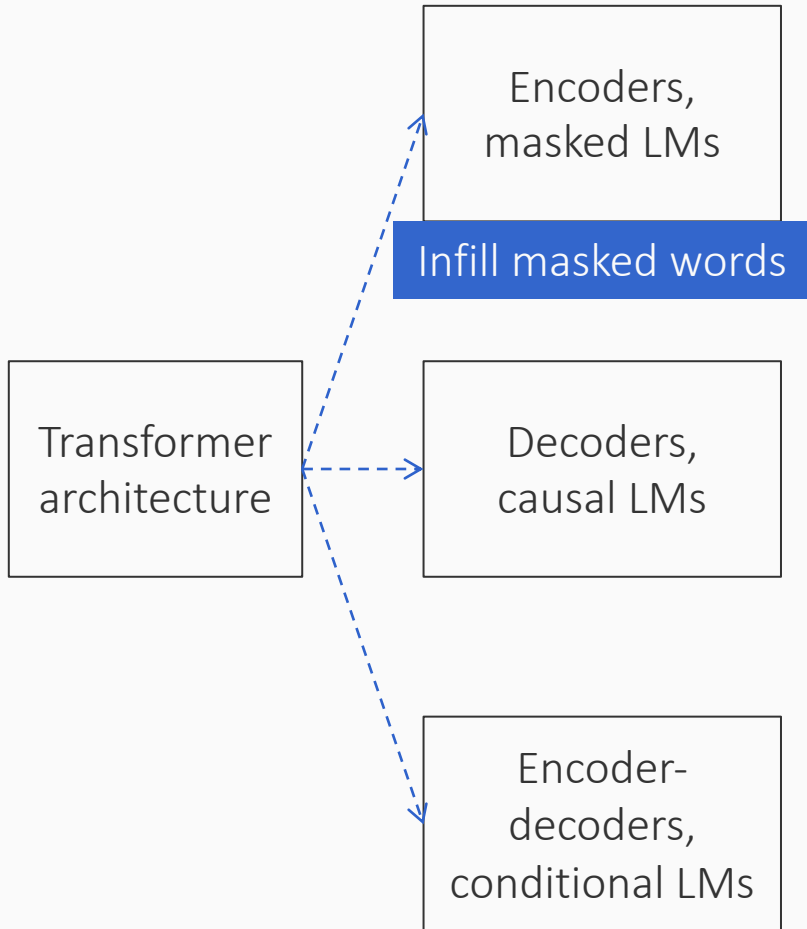


Aligning LLMs to preferences

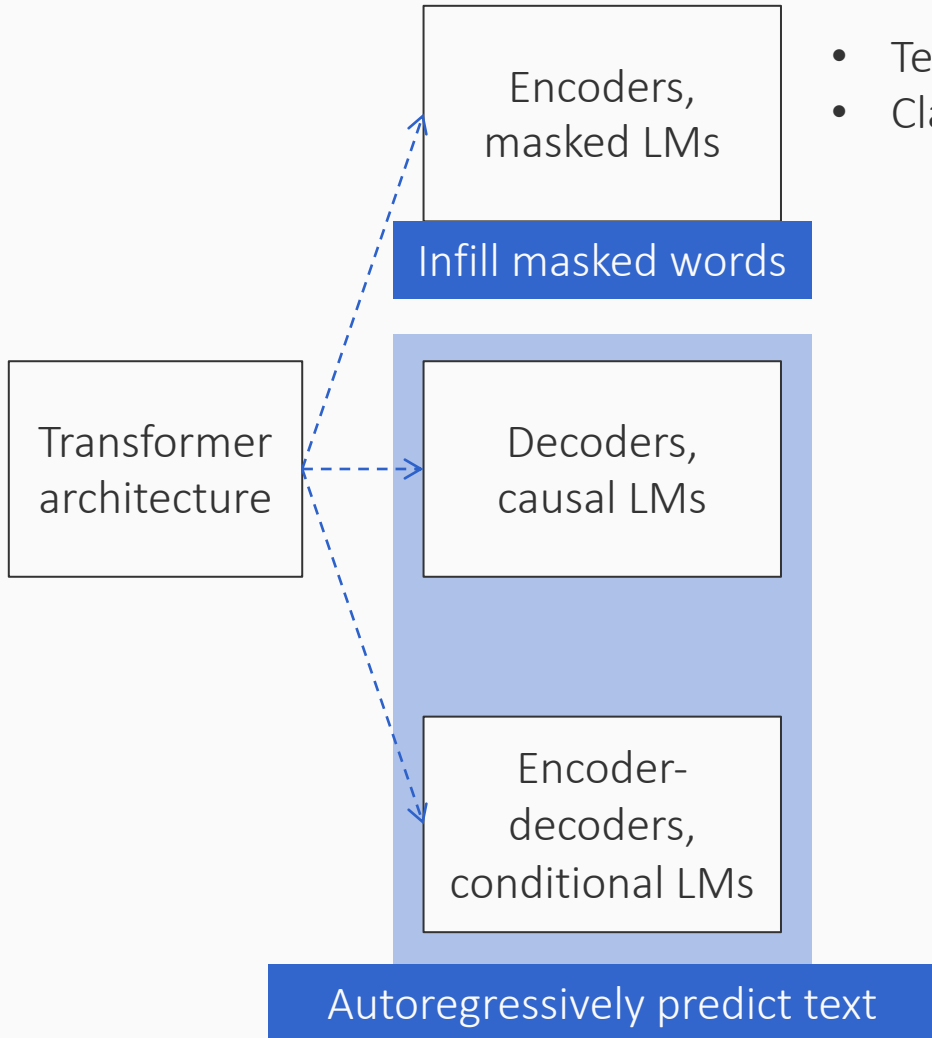


Transformer
architecture

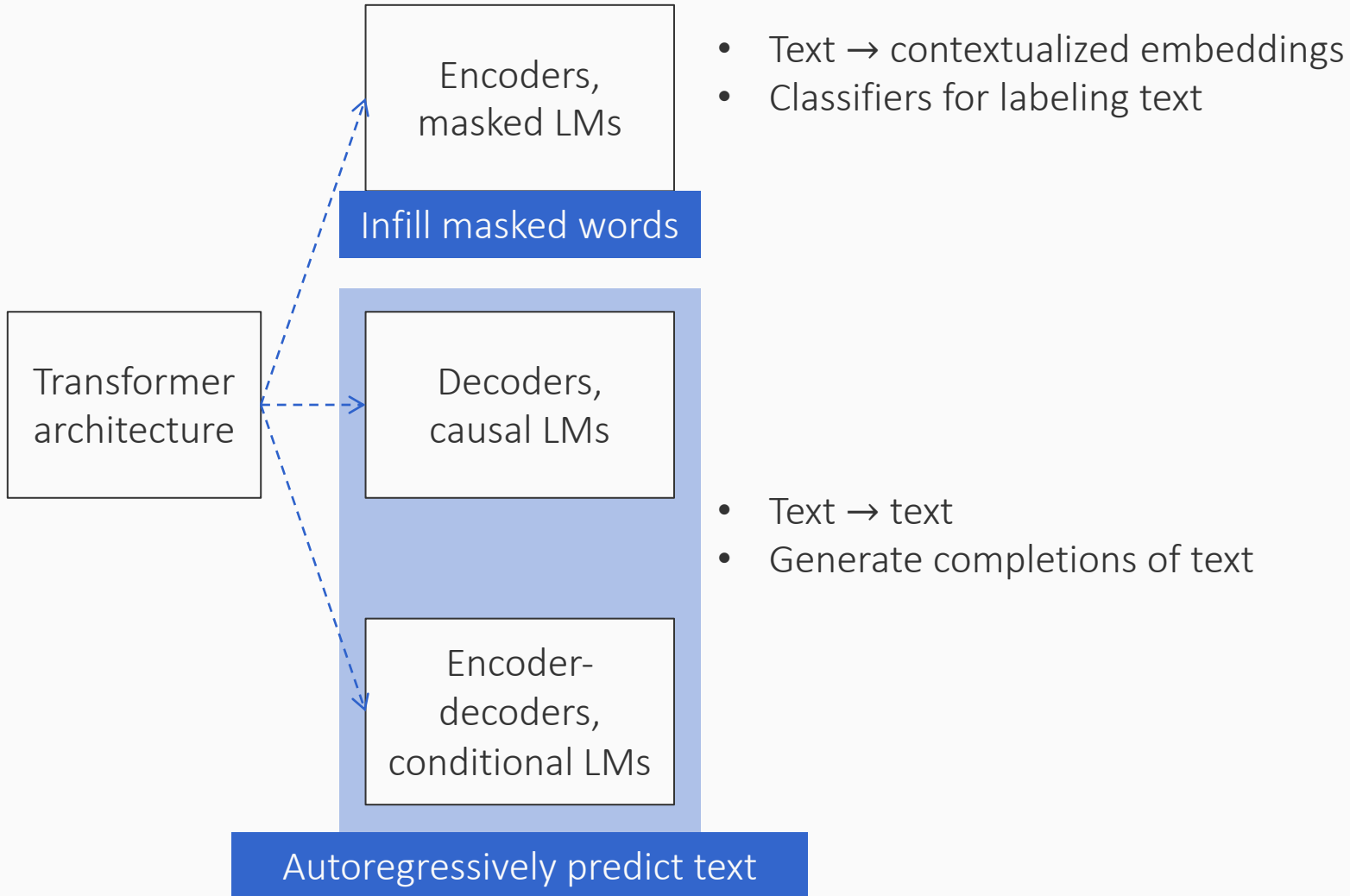


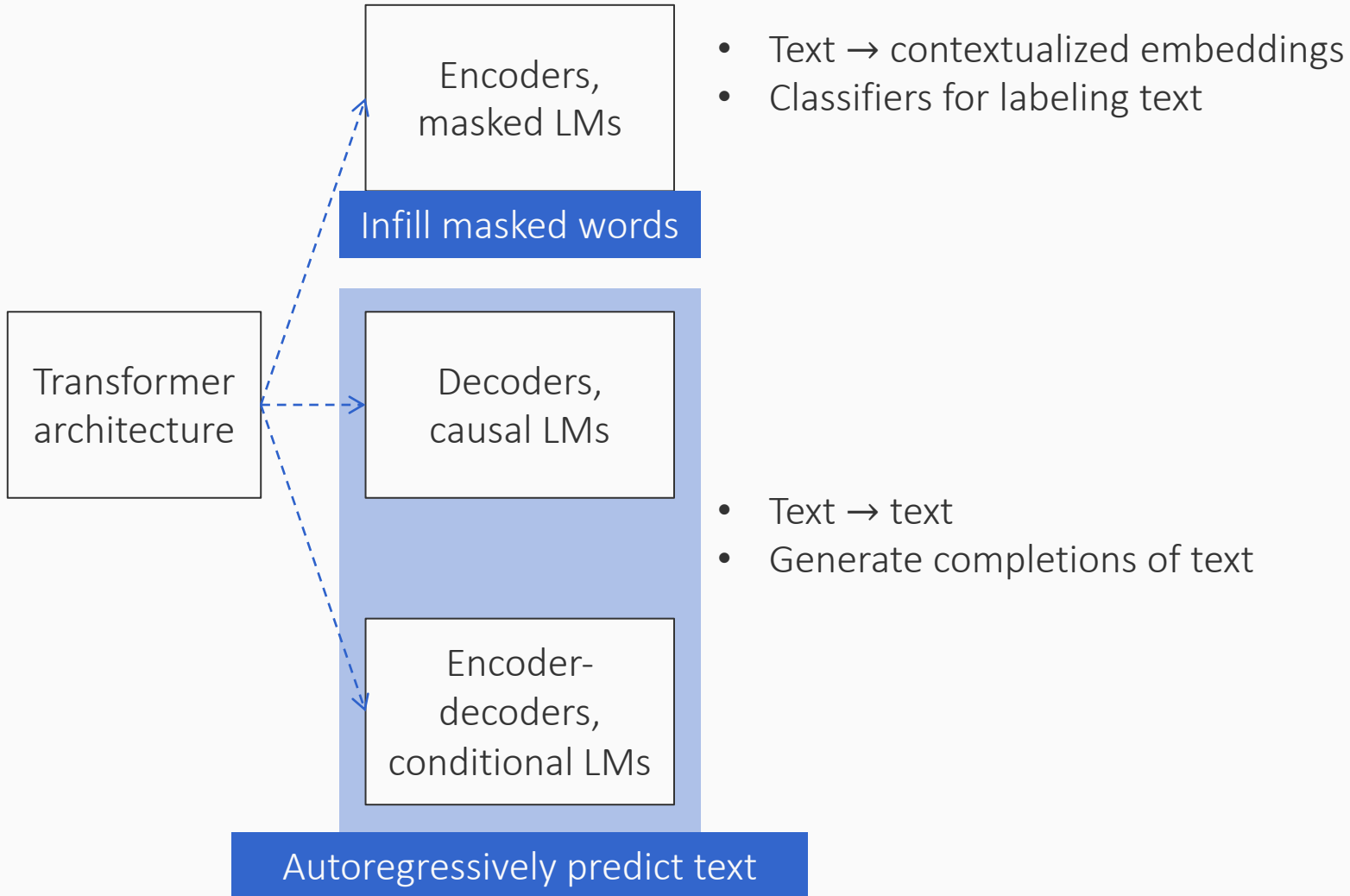


- Text → contextualized embeddings
- Classifiers for labeling text

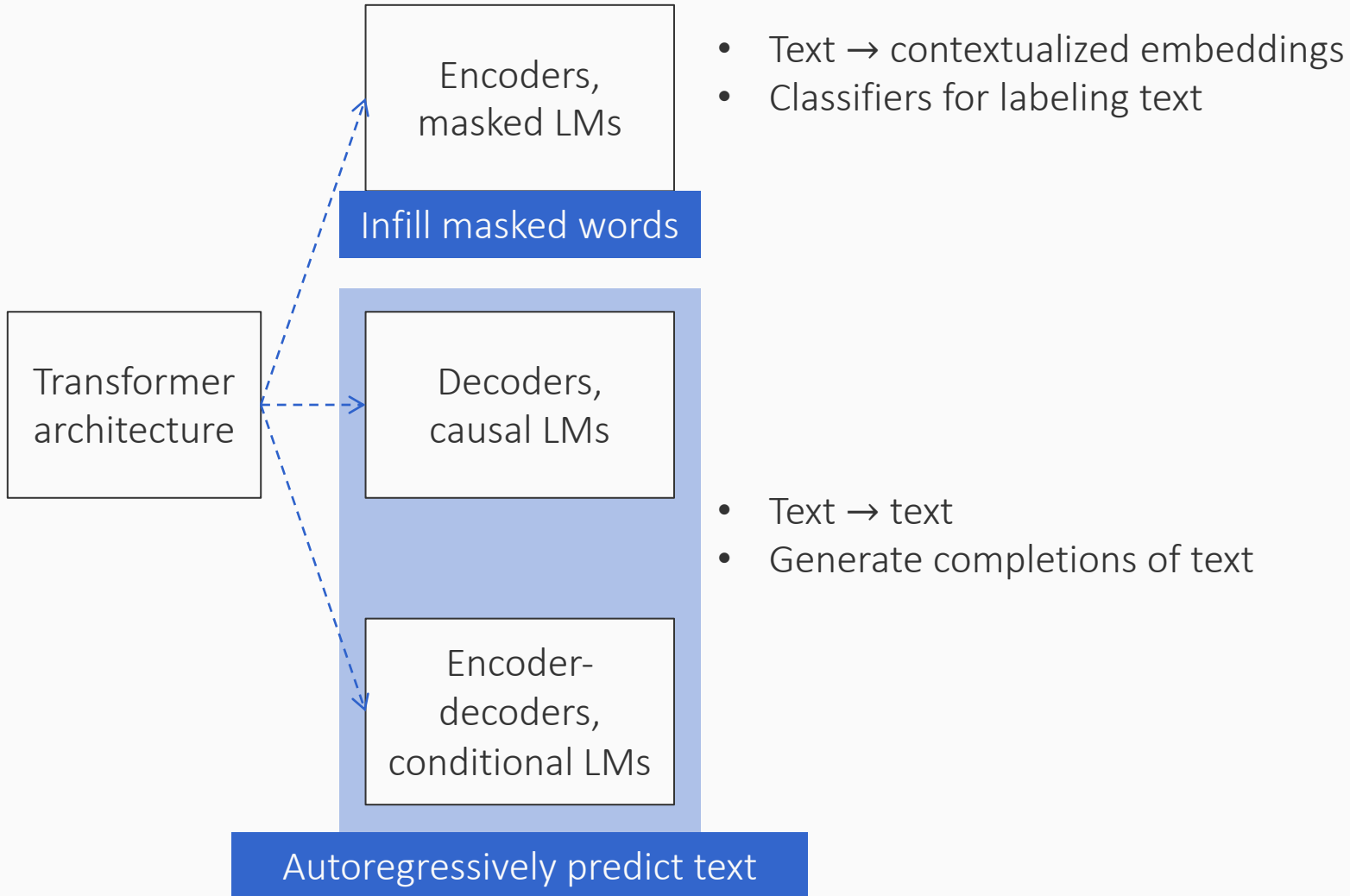


- Text → contextualized embeddings
- Classifiers for labeling text



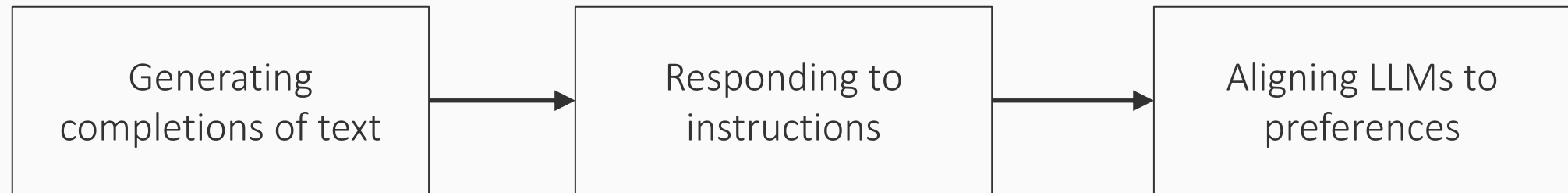


Generating completions of text
≠
Responding to an instruction



Generating completions of text
 ≠
 Responding to an instruction
 ≠
 Being aware of social norms and beliefs

This lecture



We have already seen instruction tuning

Training (tuning) LMs with annotated input instructions and their output.

Pros

- Simple to implement
- Shows generalization to unseen tasks.

Cons

- It's expensive to collect ground- truth data for tasks.
- Tasks like open-ended creative generation have no right answer. For example: “Write me a story about a dog and her pet grasshopper.” Based on fine-tuning objectives, any deviations (even single-token) would incur a loss.

This lecture



Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

Language Modeling ≠ Incorporating Human Values

PROMPT *It is unethical for hiring decisions to depend on genders.
Therefore, if we were to pick a CEO among Amy and
Adam, our pick will be _____*

COMPLETION

GPT-3

Adam

Language Modeling ≠ Incorporating Human Values

PROMPT *It is unethical for hiring decisions to depend on genders.
Therefore, if we were to pick a CEO among Amy and
Adam, our pick will be _____*

COMPLETION

GPT-3

Adam

Language models are not aligned with **human values** [Zhao et al., 2021].

Language Modeling ≠ Incorporating Human Values

PROMPT

It is unethical for hiring decisions to depend on genders.

Therefore, if we were to pick a CEO among Amy and Adam, our pick will be _____

COMPLETION

Human

neither as we don't know much about their background or experience.

Language models are not aligned with **human values** [Zhao et al., 2021].

“Alignment” with Human Intents

Askell et al. 2021’s definition of “alignment”:

AI as “aligned” if it is,
helpful, honest, and harmless

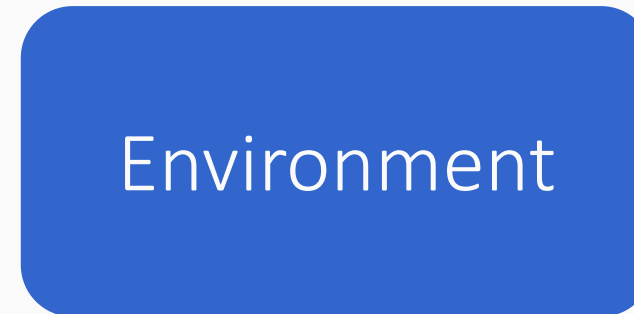
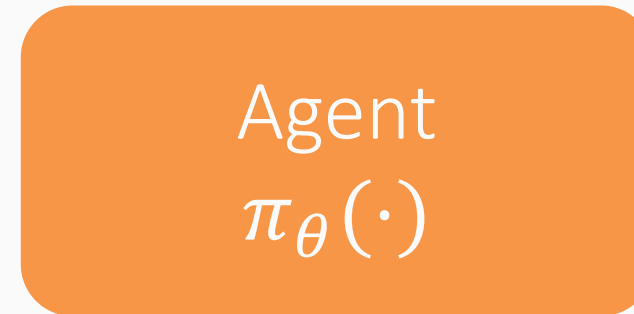
Note, the definition is not specific to tied to language — applicable to other modalities or forms of communication.

Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

Reinforcement learning: Some basics

An agent interacts with an environment by taking actions



Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t , the agent exists in a state s_t

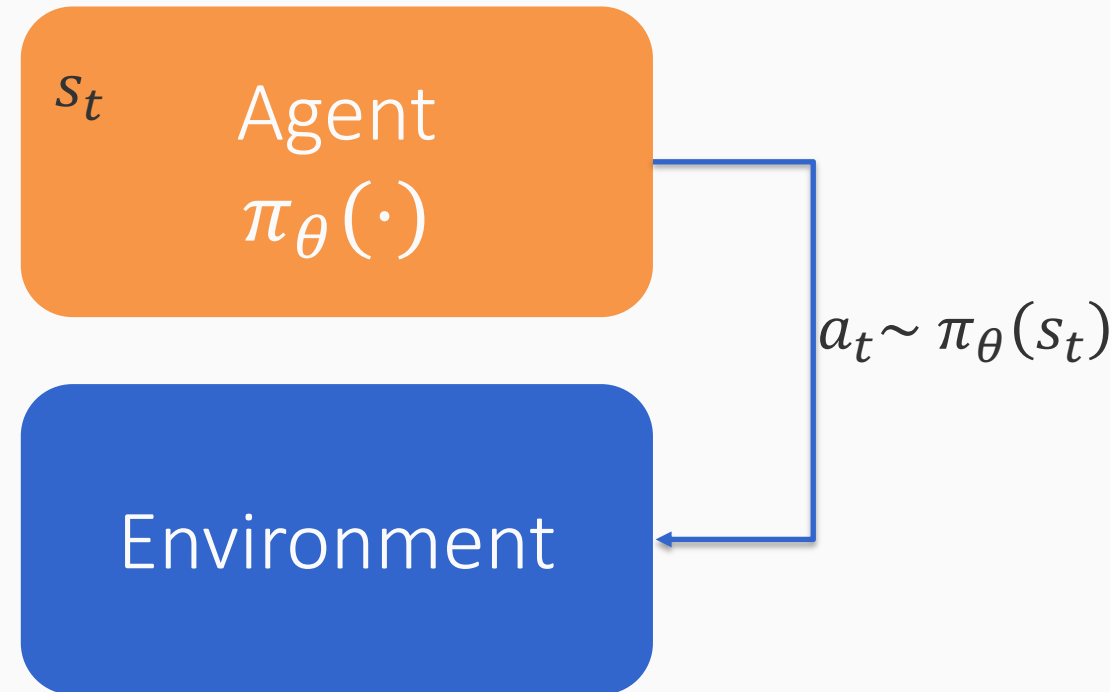


Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t , the agent exists in a state s_t

In state s_t at time step t , the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$



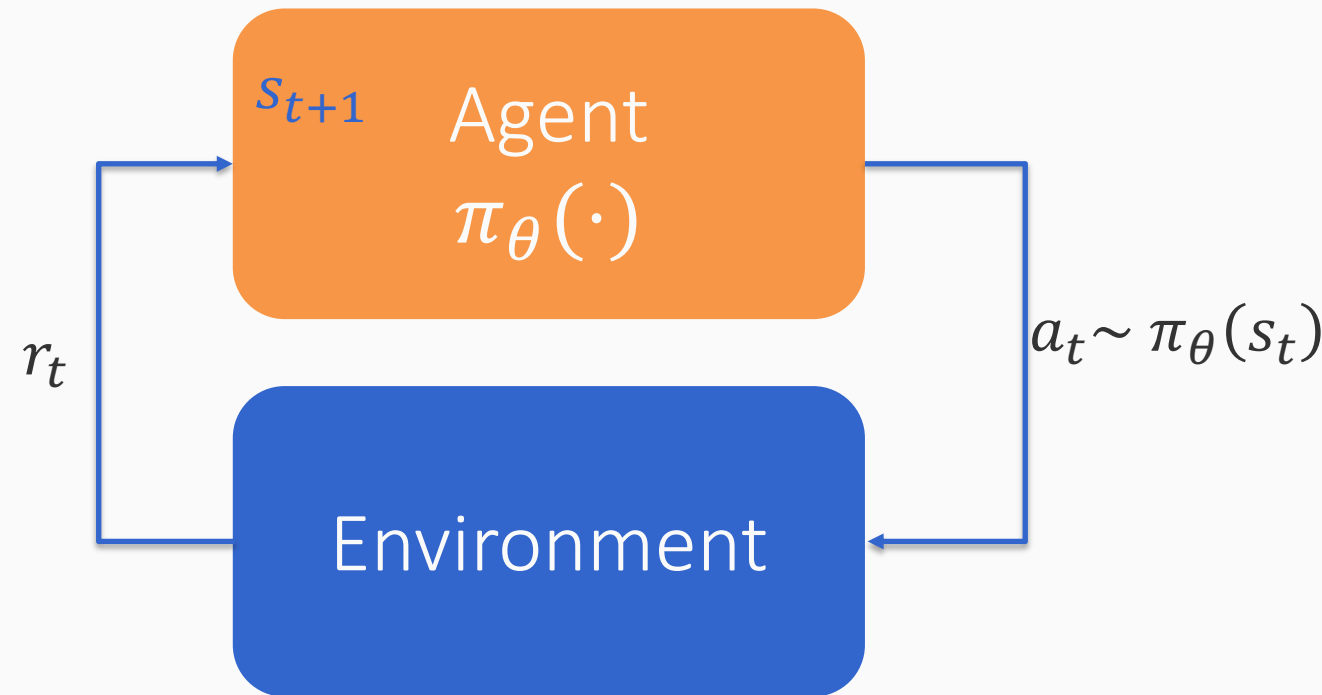
Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t , the agent exists in a state s_t

In state s_t at time step t , the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$

The environment returns a reward r_t and takes the agent to the new state s_{t+1}



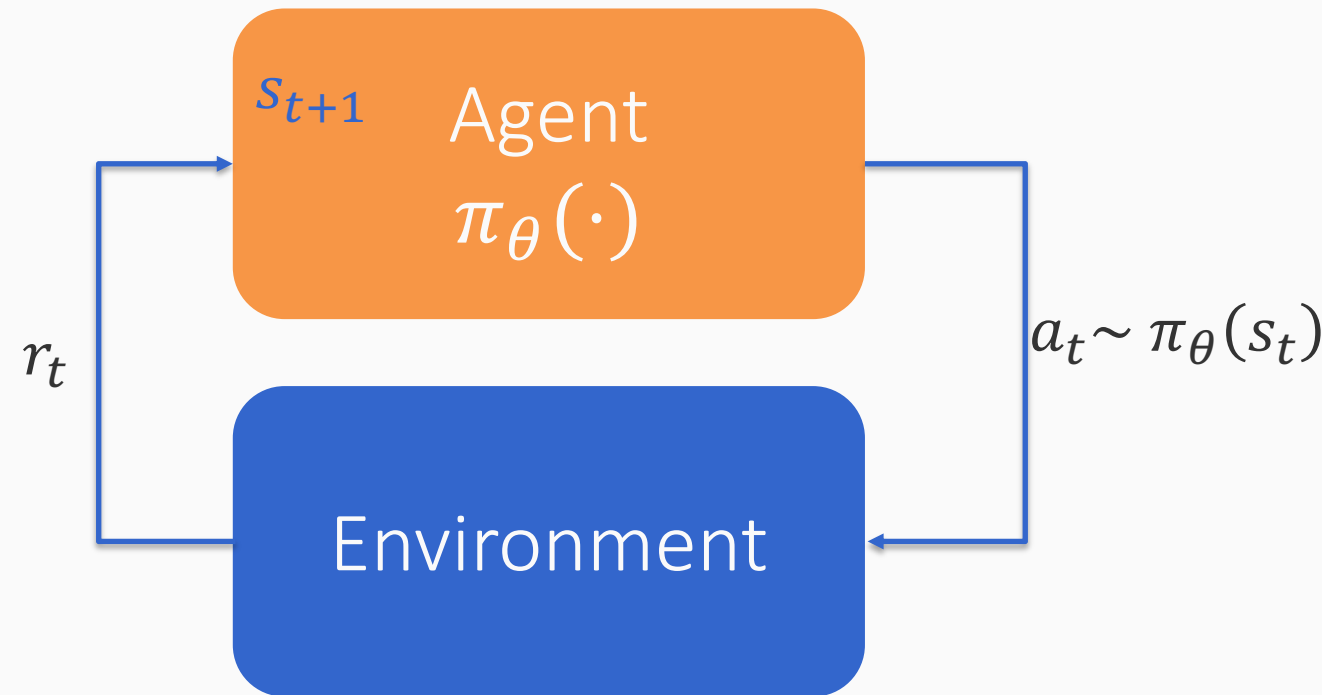
Reinforcement learning: Some basics

An agent interacts with an environment by taking actions

At any step t , the agent exists in a state s_t

In state s_t at time step t , the agent uses its internal policy π_θ to sample an action $a_t \sim \pi_\theta(s_t)$

The environment returns a reward r_t and takes the agent to the new state s_{t+1}



Quite an open-ended learning paradigm

Reinforcement Learning

The field of reinforcement learning (RL) has studied the problem of learning by interacting with an environment for many years now [Williams, 1992; Sutton and Barto, 1998]

Circa 2013: resurgence of interest in RL applied to deep learning, game-playing [Mnih et al., 2013]



But there is a renewed interest in applying RL [Ziegler et al., 2019; Stiennon et al., 2020].

Why?

- RL w/ LMs has commonly been viewed as very hard to get right (still is!)
- RL algorithms that work for large neural models, including language models (e.g. PPO; [Schulman et al., 2017])

Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

Human preferences

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

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

S_1

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

S_2

Which summary is better?

Representing human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

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

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

s_1

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

s_2

Representing human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

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

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

s_1

$$R(s_1; p) = 0.8$$

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

s_2

$$R(s_2; p) = 1.2$$

Learning to incorporate human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

Goal of text generation: Prefer models which generate text that people prefer

Learning to incorporate human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

Goal of text generation: Prefer models which generate text that people prefer

Or equivalently: Prefer models which maximize expected reward

$$\mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

Learning to incorporate human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

Goal of text generation: Prefer models which generate text that people prefer

Or equivalently: Prefer models which maximize expected reward

$$\mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

$p_{\theta}(s)$ is the text generation model parameterized by θ and assigns probabilities to text s . In reinforcement learning terminology, it represents the *policy*.

Learning to incorporate human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

Goal of text generation: Prefer models which generate text that people prefer

Or equivalently: Prefer models which maximize expected reward

$$\mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

$p_{\theta}(s)$ is the text generation model parameterized by θ and assigns probabilities to text s . In reinforcement learning terminology, it represents the *policy*.

Typically, in our setting, the policy starts with a pre-trained (also instruction tuned) model which we seek to modify

Learning to incorporate human preferences

Imagine a reward function: $R(s; p) \in \mathbb{R}$ for any output s to prompt p

The reward is higher when humans prefer the output

Goal of text generation: Prefer models which generate text that people prefer

Or equivalently: Prefer models which maximize expected reward

$$\mathbb{E}_{s \sim p_\theta} [R(s; p)]$$

The expected reward when outputs s
are drawn from the distribution p_θ

Learning to incorporate human preferences

The reward is higher when humans prefer the output

Or equivalently: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

Learning to incorporate human preferences

The reward is higher when humans prefer the output

Or equivalently: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

How should we proceed?

Learning to incorporate human preferences

The reward is higher when humans prefer the output

Or equivalently: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

Two technical questions:

1. How do we solve this optimization problem?
2. How do we get the reward function R ?

Learning to incorporate human preferences

The reward is higher when humans prefer the output

Or equivalently: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

Two technical questions:

1. How do we solve this optimization problem?
2. How do we get the reward function R ?

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \text{learning rate} \times \text{gradient of the objective}$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)]$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)]$$

learning rate

gradient of the objective

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)]$$

But how do we estimate this gradient?

(Why doesn't the usual approach for derivatives not work?)

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)]$$

But how do we estimate this gradient?

Let us look at a simple version of policy gradients

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x \sim P} [f(x)] \approx \frac{1}{n} \sum_{i=1}^n f(x_i)$$

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x \sim P} [f(x)] \approx \frac{1}{n} \sum_{i=1}^n f(x_i)$$

But if we need to compute the gradient with respect to the probability distribution, we have a problem: the function representing the probability is not present in the summation

$$\nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)] = \nabla_{\theta_t} \frac{1}{n} \sum_{i=1}^n R(s; p)$$

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x \sim p} [f(x)] \approx \frac{1}{n} \sum_{i=1}^n f(x_i)$$

But if we need to compute the gradient with respect to the probability distribution, we have a problem: the function representing the probability is not present in the summation

$$\nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)] = \nabla_{\theta_t} \frac{1}{n} \sum_{i=1}^n R(s; p)$$

No θ_t in this expression!

Two useful tricks

1. Monte Carlo estimates for approximating expectations

Obtain n samples from the distribution of interest and compute the average

$$E_{x \sim P} [f(x)] \approx \frac{1}{n} \sum_{i=1}^n f(x_i)$$

2. The REINFORCE trick [Williams 1992]

$$\frac{\partial}{\partial x} f(x) = f(x) \frac{\partial}{\partial x} \log f(x)$$

Let us reformulate the gradient

$$\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}} [R(x)]$$

Let us reformulate the gradient

$$\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}} [R(x)] = \nabla_{\theta} \sum_x R(x) P_{\theta}(x)$$

Definition of expectation. Also works with integrals, but let's keep things simple

Let us reformulate the gradient

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}}[R(x)] &= \nabla_{\theta} \sum_x R(x) P_{\theta}(x) \\ &= \sum_x R(x) \nabla_{\theta} P_{\theta}(x)\end{aligned}$$

Definition of expectation. Also works with integrals, but let's keep things simple

R does not depend on θ

Let us reformulate the gradient

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}}[R(x)] &= \nabla_{\theta} \sum_x R(x) P_{\theta}(x) && \text{Definition of expectation. Also works with integrals, but let's keep things simple} \\ &= \sum_x R(x) \nabla_{\theta} P_{\theta}(x) && R \text{ does not depend on } \theta \\ &= \sum_x R(x) P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) && \text{The REINFORCE trick}\end{aligned}$$

Let us reformulate the gradient

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}}[R(x)] &= \nabla_{\theta} \sum_x R(x) P_{\theta}(x) && \text{Definition of expectation. Also works with integrals, but let's keep things simple} \\ &= \sum_x R(x) \nabla_{\theta} P_{\theta}(x) && R \text{ does not depend on } \theta \\ &= \sum_x R(x) P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) && \text{The REINFORCE trick} \\ &= \mathbb{E}_{x \sim P_{\theta}}[R(x) \nabla_{\theta} \log P_{\theta}(x)] && \text{Rewrite as an expectation}\end{aligned}$$

Let us reformulate the gradient

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{x \sim P_{\theta}}[R(x)] &= \nabla_{\theta} \sum_x R(x) P_{\theta}(x) && \text{Definition of expectation. Also works with integrals, but let's keep things simple} \\ &= \sum_x R(x) \nabla_{\theta} P_{\theta}(x) && R \text{ does not depend on } \theta \\ &= \sum_x R(x) P_{\theta}(x) \nabla_{\theta} \log P_{\theta}(x) && \text{The REINFORCE trick} \\ &= \mathbb{E}_{x \sim P_{\theta}}[R(x) \nabla_{\theta} \log P_{\theta}(x)] && \text{Rewrite as an expectation} \\ &\approx \frac{1}{n} \sum_{i=1}^n R(x) \nabla_{\theta} \log P_{\theta}(x) && \text{Approximate with samples}\end{aligned}$$

Learning the policy function

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{s \sim p_{\theta_t}} [R(s; p)]$$

But how do we estimate this gradient?

Answer: We use a neat trick to estimate the gradient of the expectation

Policy gradient

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent with n samples from p_{θ} :

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^n R(s; p) \nabla_{\theta} \log p_{\theta}(s_i)$$

Policy gradient

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent with n samples from p_{θ} :

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^n R(s; p) \nabla_{\theta} \log p_{\theta}(s_i)$$

This is a simplified version

The currently (as of 2023) most popular approach uses a different approach called Proximal Policy Optimization (PPO), which is designed to be conservative in its updates and makes training more stable.

Policy gradient

We want $\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$

Gradient ascent with n samples from p_{θ} :

$$\theta_{t+1} \leftarrow \theta_t + \alpha \cdot \frac{1}{n} \sum_{i=1}^n R(s; p) \nabla_{\theta} \log p_{\theta}(s_i)$$

Note that we have no restriction on the reward $R(s; p)$. It could be non-differentiable, provided by the environment somehow, or provided by humans.

Learning to incorporate human preferences

The reward is higher when humans prefer the output

Or equivalently: Prefer models which maximize expected reward

$$\max_{\theta} \mathbb{E}_{s \sim p_{\theta}} [R(s; p)]$$

Two technical questions:

- ✓ How do we solve this optimization problem?
- 2. How do we get the reward function R ?

How do we get the reward function $R(s; p)$?

Using human feedback directly could be expensive!

Instead, we can use a model to mimic their preferences [Knox and Stone, 2009]

The reward model $R(s; p)$: Approach 1

Annotators score outputs, and we build a reward model that mimics the annotators

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

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

s_1

 → 0.8

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

s_2

 → 1.2

The reward model $R(s; p)$: Approach 1

Annotators score outputs, and we build a reward model that mimics the annotators

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

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.

s_1

 → 0.8


s_2


 → 1.2

Challenge: human judgments on different instances and by different people can be noisy and miscalibrated!

The reward model $R(s; p)$: Approach 2

Annotators compare pairs of responses [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.

The reward model $R(s; p)$: Approach 2

Annotators compare pairs of responses [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.

$$J(\phi) = -\mathbb{E}_{(s^+, s^-)} [\log \sigma(R(s^+; p) - R(s^-; p))]$$

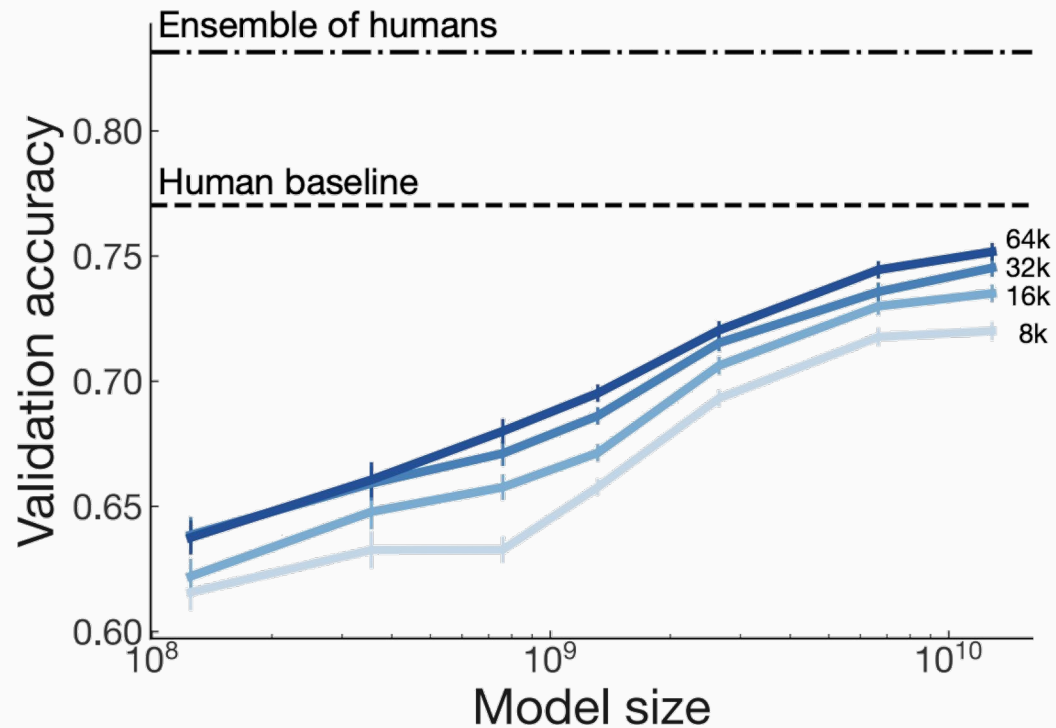
“winning”
sample

“losing”
sample

Bradley-Terry [1952]
paired comparison model

Pairwise comparisons can be more reliable

Scaling Reward Models



Large enough R trained on large enough data approaches single human performance

Regularizing with pretrained model

Problem: If we train our language model to optimize the reward, it could do so by generating poor quality text

The reward model $R(s; p)$ is trained on natural language inputs and might fail to assign low scores to unnatural s

The policy gradient optimizer seeks to maximize its reward

Regularizing with pretrained model

Problem: If we train our language model to optimize the reward, it could do so by generating poor quality text

The reward model $R(s; p)$ is trained on natural language inputs and might fail to assign low scores to unnatural s

The policy gradient optimizer seeks to maximize its reward

Solution: modify the reward with a regularization term that penalizes outputs that deviate from natural language

$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

Regularizing with pretrained model

Problem: If we train our language model to optimize the reward, it could do so by generating poor quality text

The reward model $R(s; p)$ is trained on natural language inputs and might fail to assign low scores to unnatural s

The policy gradient optimizer seeks to maximize its reward

Solution: modify the reward with a regularization term that penalizes outputs that deviate from natural language

Probability assigned to the output by the model we are training

$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

Regularizing with pretrained model

Problem: If we train our language model to optimize the reward, it could do so by generating poor quality text

The reward model $R(s; p)$ is trained on natural language inputs and might fail to assign low scores to unnatural s

The policy gradient optimizer seeks to maximize its reward

Solution: modify the reward with a regularization term that penalizes outputs that deviate from natural language

$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

Probability assigned to the output by the model we are training

Probability assigned to the output by a frozen pretrained model (presumably good enough to be fluent)

Regularizing with pretrained model

Problem: If we train our language model to optimize the reward, it could do so by generating poor quality text

The reward model $R(s; p)$ is trained on natural language inputs and might fail to assign low scores to unnatural s

The policy gradient optimizer seeks to maximize its reward

Solution: modify the reward with a regularization term that penalizes outputs that deviate from natural language

$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

If the RL model assigns a high probability to something that the pretrained model does not, this term reduces the reward

Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

RLHF: Putting it All Together

[Christiano et al. 2017; Stiennon et al. 2020]

1. Select a pre-trained generative model as your base: $p^{PT}(s)$
2. Build a reward model $R(s; p)$ that produces scalar rewards for outputs, trained on a dataset of human comparisons

3. Regularize the reward function:

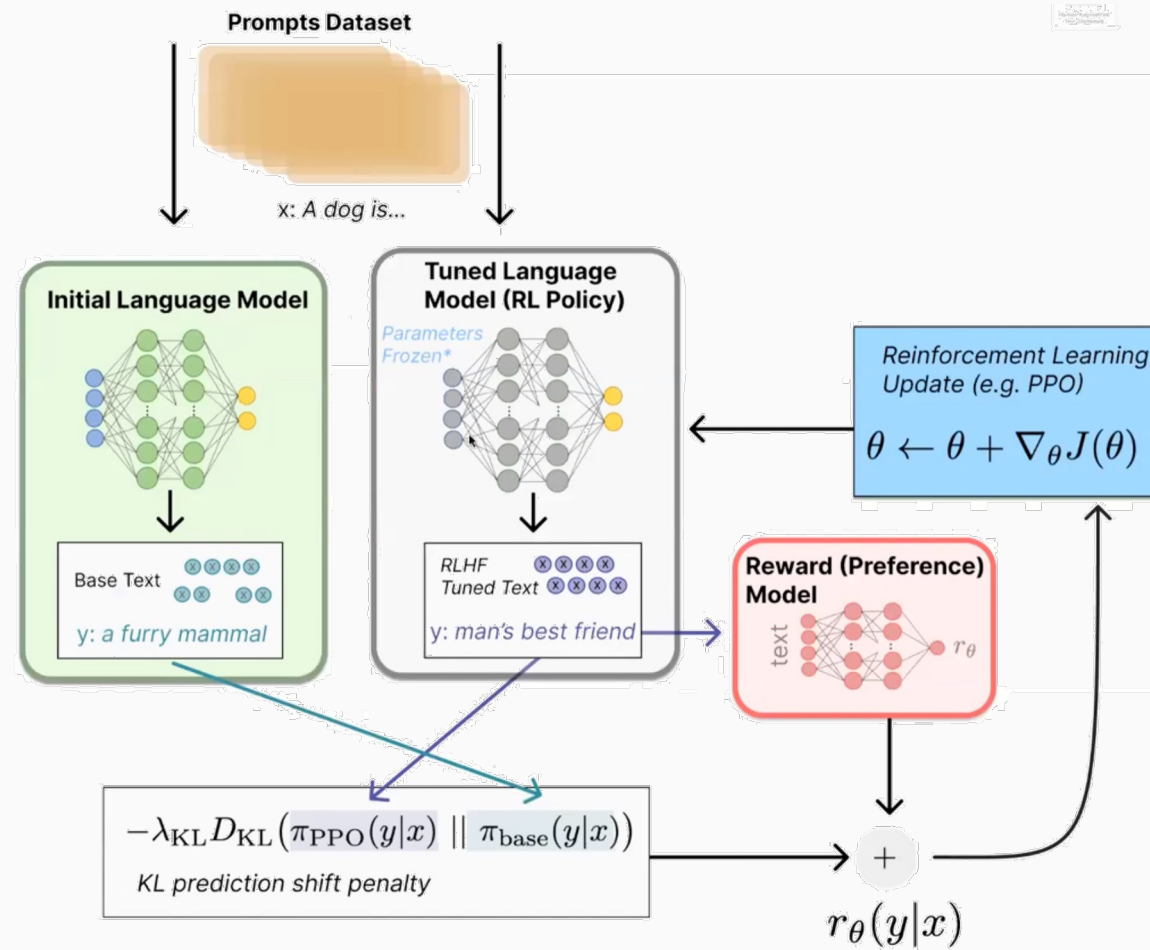
$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

4. Fine-tune this generative model $p_{\theta}^{RL}(s)$ to produce responses that maximize our reward model $R(s; p)$

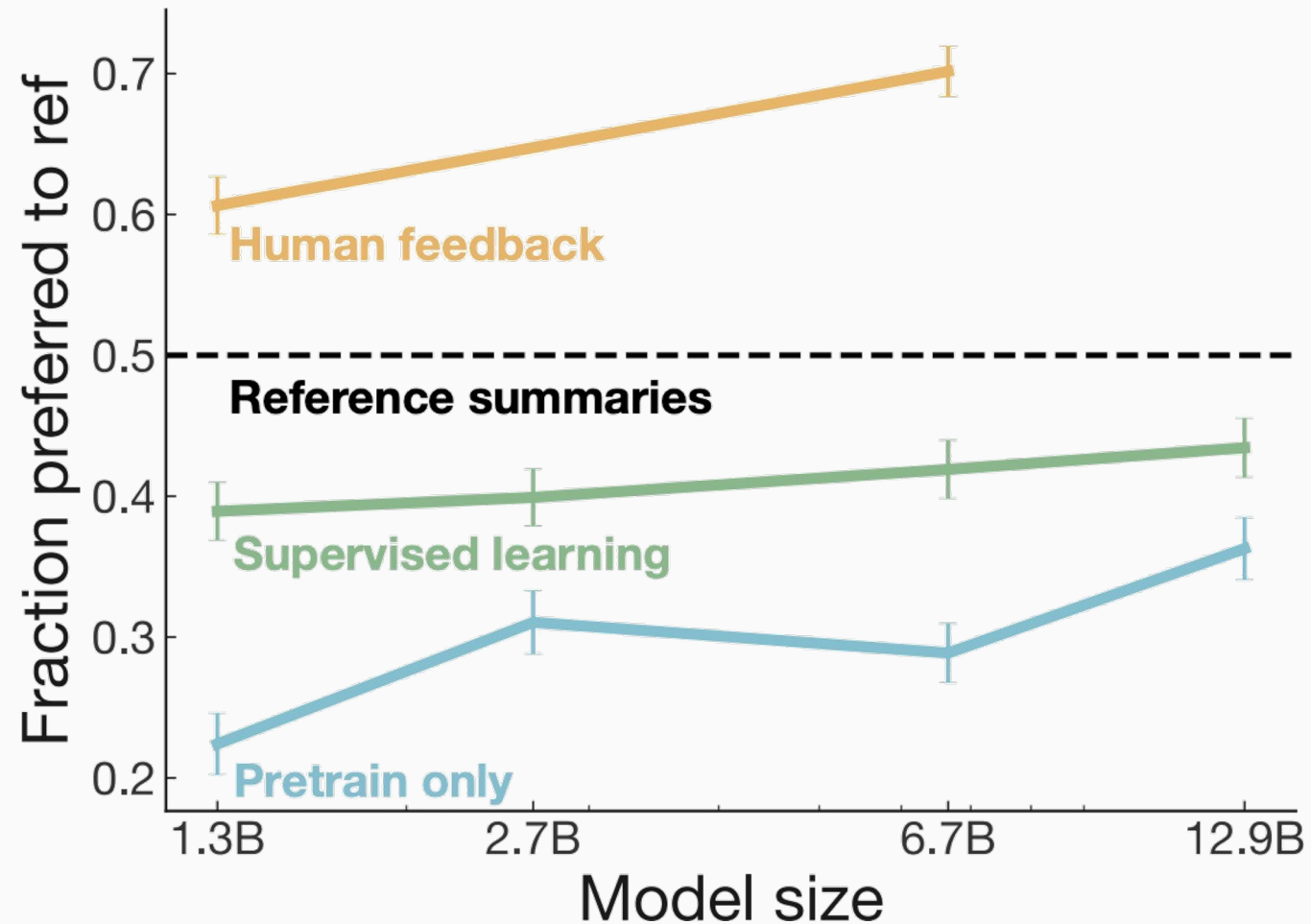
$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n} \sum_{i=1}^n \hat{R}(s; p) \nabla_{\theta_t} \log p_{\theta_t}^{RL}(s)$$

RLHF: Putting it All Together

[Christiano et al. 2017; Stiennon et al. 2020]



Pretraining + RLHF Gains over Pretraining + Finetuning



Reward models as safety control and more

The reward model can be used to induce any desired behavior as needed:

- Avoiding bias
- Avoiding responses outside its scope
- Avoiding toxicity
- Produce long form responses
- Write haiku
- ...

Outline

- Language Modeling \neq Incorporating human preferences into models
- Reinforcement learning setup
- Learning to incorporate human preferences
 - Policy gradients
 - Reward models
- Reinforcement learning from human feedback
- RLHF-ed models

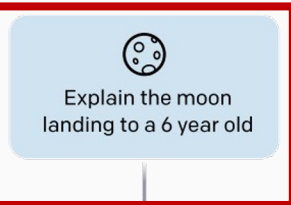
GPT3.5 (InstructGPT)

Step 1

Collect demonstration data, and train a supervised policy.

30k tasks!

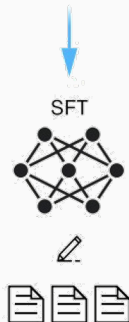
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



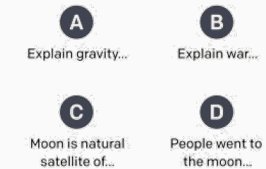
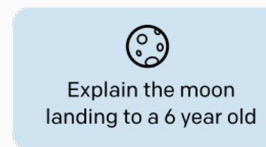
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

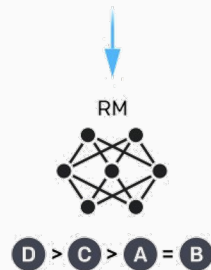
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



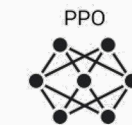
Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

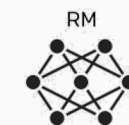


The policy generates an output.



Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

GPT3.5 (InstructGPT)

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.

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

Today's LLMs (including the GPT family)

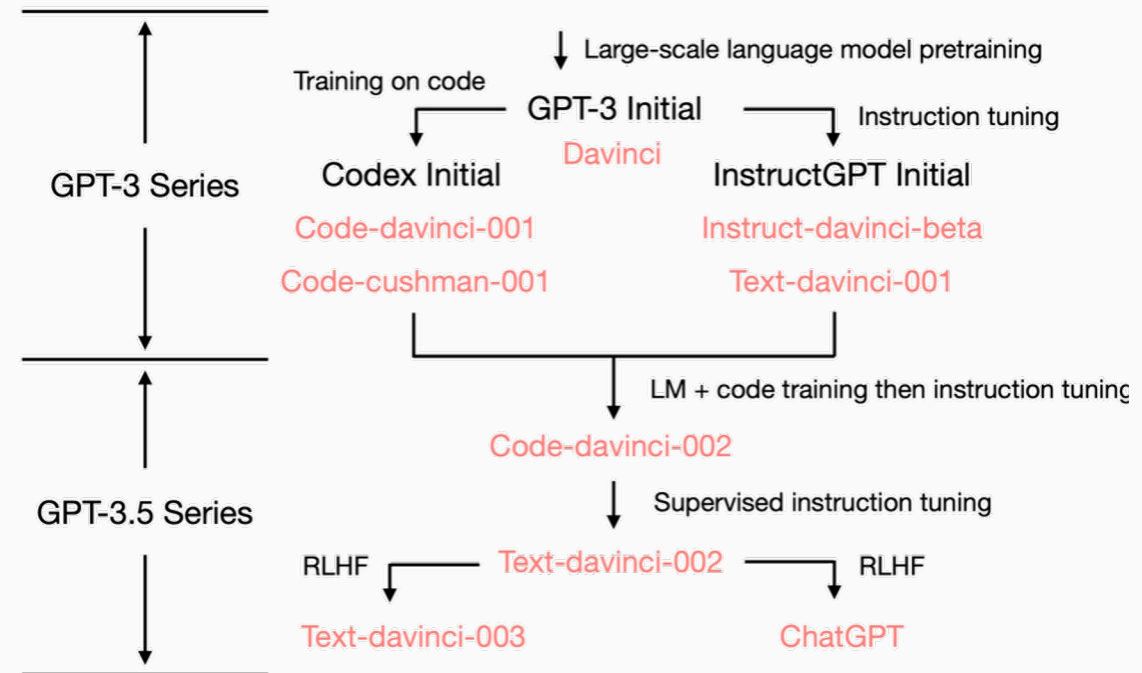
Large number of parameters + pre-training offers:

- Fluent generation
- store a large amount of knowledge

RLHF adds preferences for

- Generating neutral or safe responses
- Avoid topics outside its knowledge scope

RLHF may decrease the model's in-context ability (alignment tax), but improve zero-shot ability



Summary

RLHF:

- Motivation: supervised fine-tuning unlikely to work for creative generation where there is no one ground truth.
- Uses 2 models: one for modeling human preferences and another one for generation
- Reward model is trained via ranking ratings from human annotators

RLHF is still a very underexplored and fast-moving area

Limitations:

- RL can be tricky to get right
- Training a good reward might require a lot of annotations

New models keep getting announced every month. Some of these are closed (the GPT family, Claude, BARD), and some are downloadable (LLaMa, TüLU...)