ml-mipt-advanced

Lecture 12. LLM Alignment.

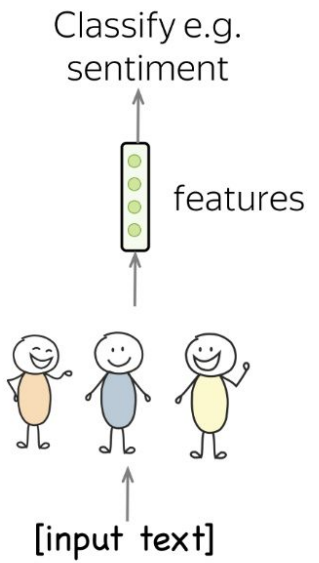
Nikolay Karpachev
22.04.2024

Outline

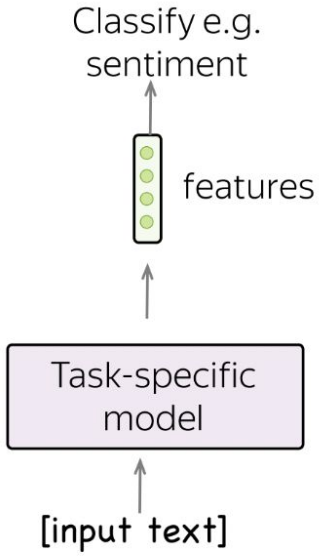
1. LLMs as instruct models
2. Finetuning for instructions
3. Alignment with user feedback
    a. RLHF
    b. Contrastive Learning

Evolutionary journey in NLP

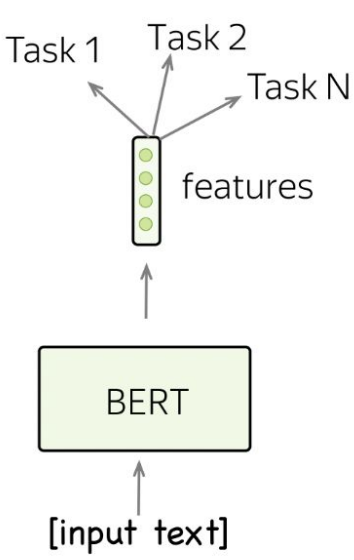
Different task – another set of features (and people!)



Different task – another model



One model, classify anything



Talk to the model

Input (prompt)

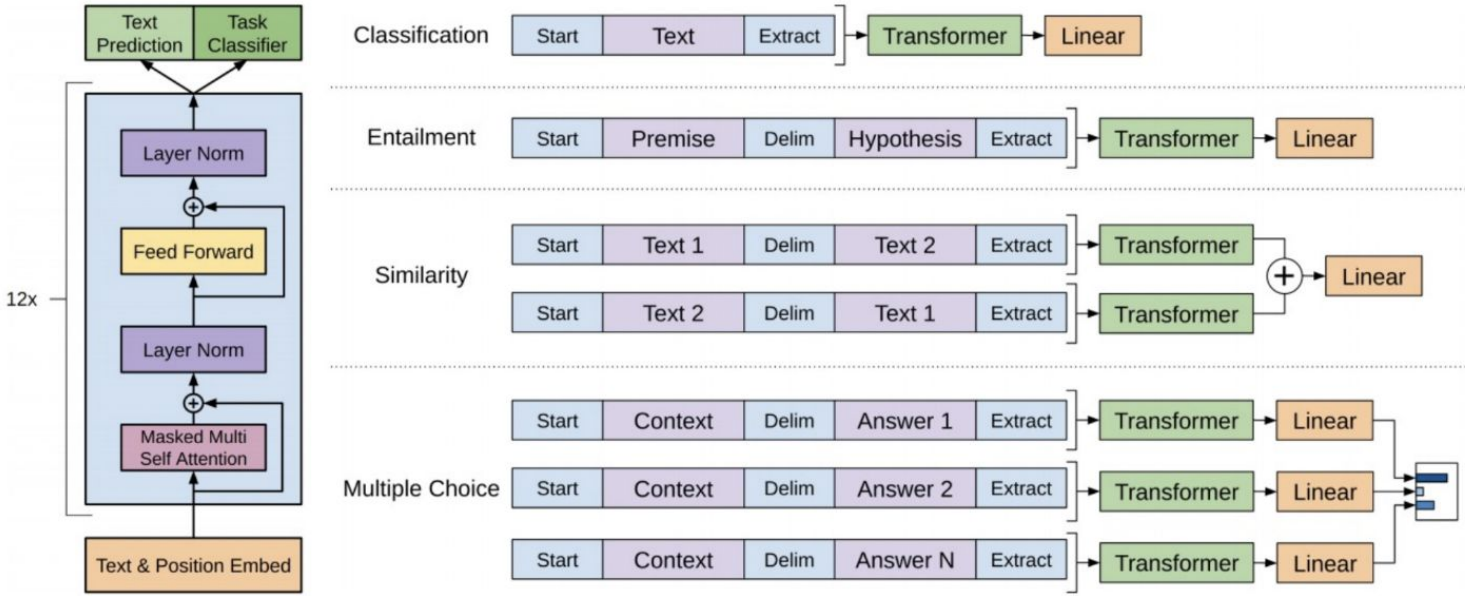
What is the sentiment of the next sentence? I love this movie!

Model output

positive

GPT

Fine-tuning loss: $L = L_{xent} + \lambda \cdot L_{task}$



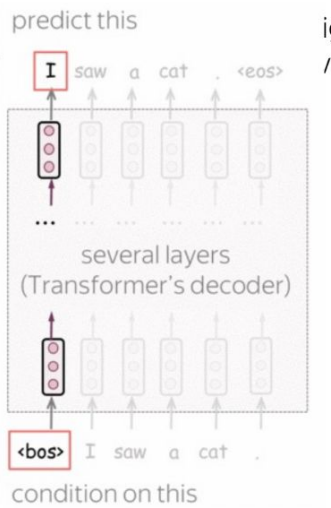
The figure is from the paper Improving Language Understanding by Generative Pretraining

GPT

Training

Transformer decoder with the standard l

lar



ight
/e

Inference

GPT-1 Classification via

- Task-specific input transformations
- Supervised fine-tuning

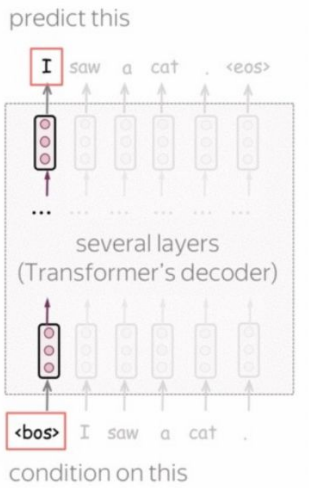
GPT

	GPT-1 (2018)	GPT-2 (2019)
Number of parameters	117 million	1.5 billion
Training data	5 GB	40 GB

GPT

Training

Transformer decoder with the standard left-to-right language modeling objective



Inference

GPT-1 Classification via
- Task-specific input transformations
- Supervised fine-tuning

GPT-2 Generation tasks via task-specific input transformations

For example, for text summarization simply add "**TL;DR**":

<start> Generative pre-trained transformers (GPT) are … TL;DR:

Text to be summarized

GPT

	GPT-1 (2018)	GPT-2 (2019)	GPT-3 (2020)
Number of parameters	117 million	1.5 billion	175 billion
Training data	5 GB	40 GB	45 TB (i.e., 45 000 GB)

The entirety of English Wikipedia constitutes just 0.6% of GPT-3 training data

GPT

GPT-3: Examples

Prompt

```
# Python 3
def remove_common_prefix(x, prefix, ws_prefix):
  x["completion"] = x["completion"].str[len(prefix) :]
  if ws_prefix:
    # keep the single whitespace as prefix
    x["completion"] = " " + x["completion"]
return x

# Explanation of what the code does

#
```

Sample response

The code above is a function that takes a dataframe and a prefix as input and returns a dataframe with the prefix removed from the completion column.

GPT

GPT-3: Examples

Create study notes

Generation

Provide a topic and get study notes.

Prompt

What are 5 key points I should know when studying Ancient Rome?

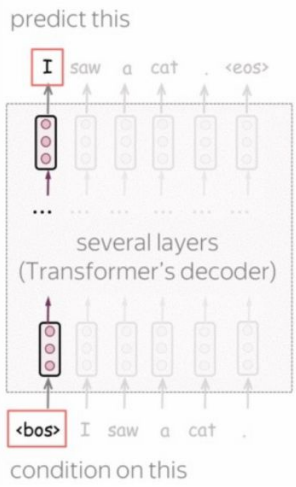
Sample response

1. Rome was founded in 753 BC by Romulus and Remus.
2. Rome was ruled by an emperor.
3. Rome was a republic before it became an empire.
4. Julius Caesar was assassinated in 44 BC.
5. Augustus was the first emperor of Rome.

GPT

Training

Transformer decoder with the standard left-to-right language modeling objective



Inference

GPT-1 Classification via
- Task-specific input transformations
- Supervised fine-tuning

GPT-2 Generation tasks via task-specific input transformations

For example, for text summarization simply add "**TL;DR**":

GPT-3 Complex generation and reasoning tasks via in-context learning: prompt with task description and a few demonstrations

```
Translate English to Spanish:
a black cat -> un gato negro
I am hungry -> tengo hambre
a cup of tea ->
```

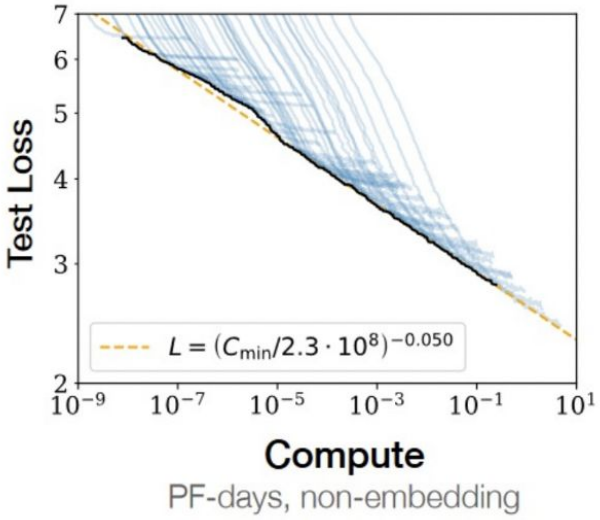
← task description

← examples

← prompt

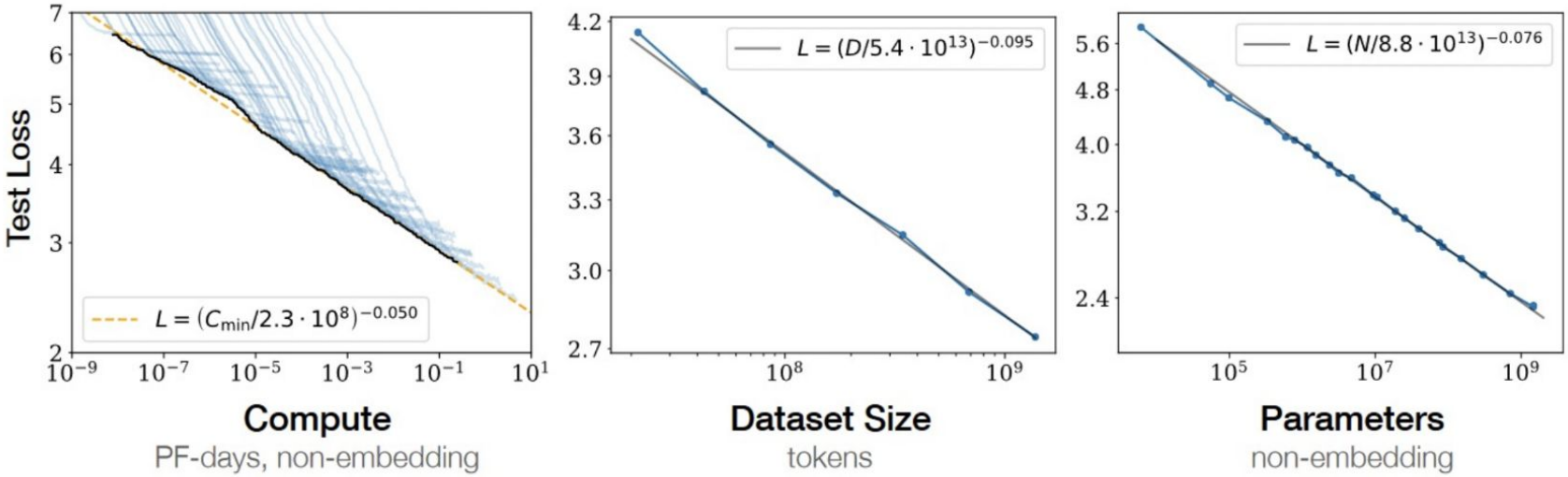
Scaling laws of LLMs

How does model quality change with respect to compute spent?



Scaling laws of LLMs

How does model quality change with respect to compute spent? | dataset size? | model size?

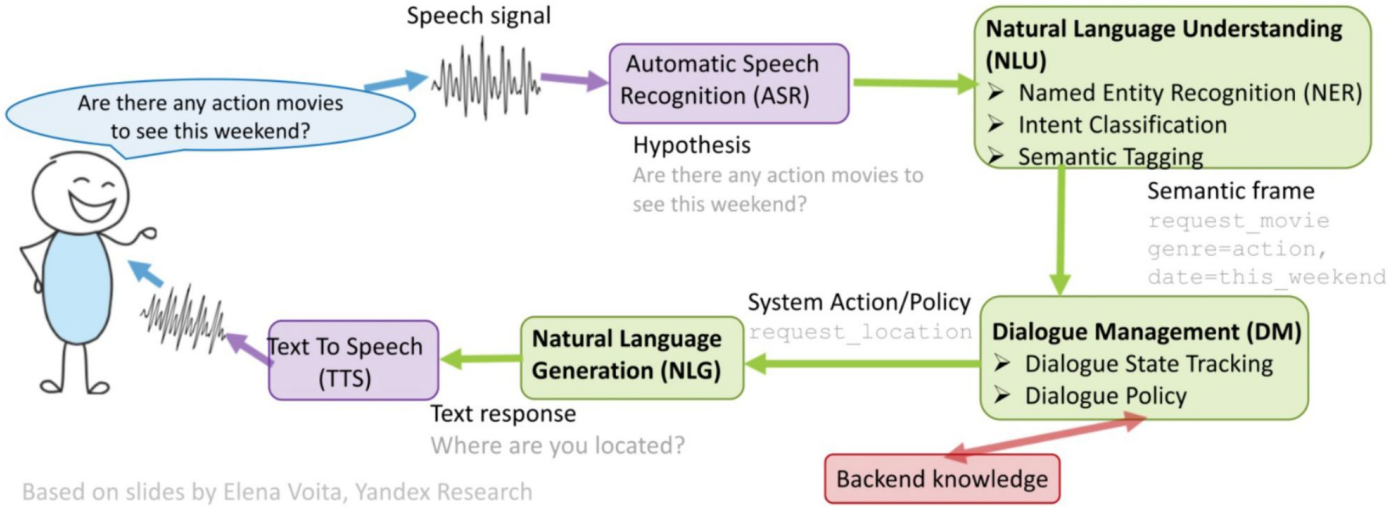


LLMs as dialogue assistants

Dialogue assistants: Cascade pipeline

System design: let's split "chat bot" into smaller problems

Speech processing: see YSDA speech course
Business knowledge: rules for banking / psychology / ...
Text processing: this lecture



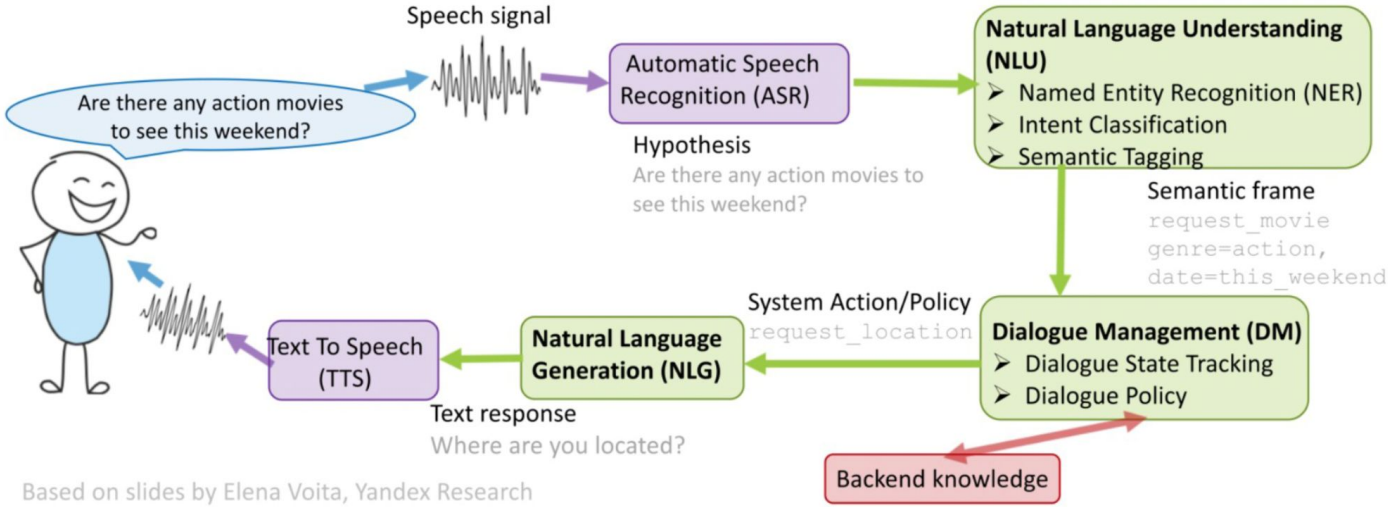
Based on slides by Elena Voita, Yandex Research

Dialogue assistants: Cascade pipeline

Any problems here?

System design: let's split "chat bot" into smaller problems

Speech processing: see YSDA speech course
Business knowledge: rules for banking / psychology / ...
Text processing: this lecture

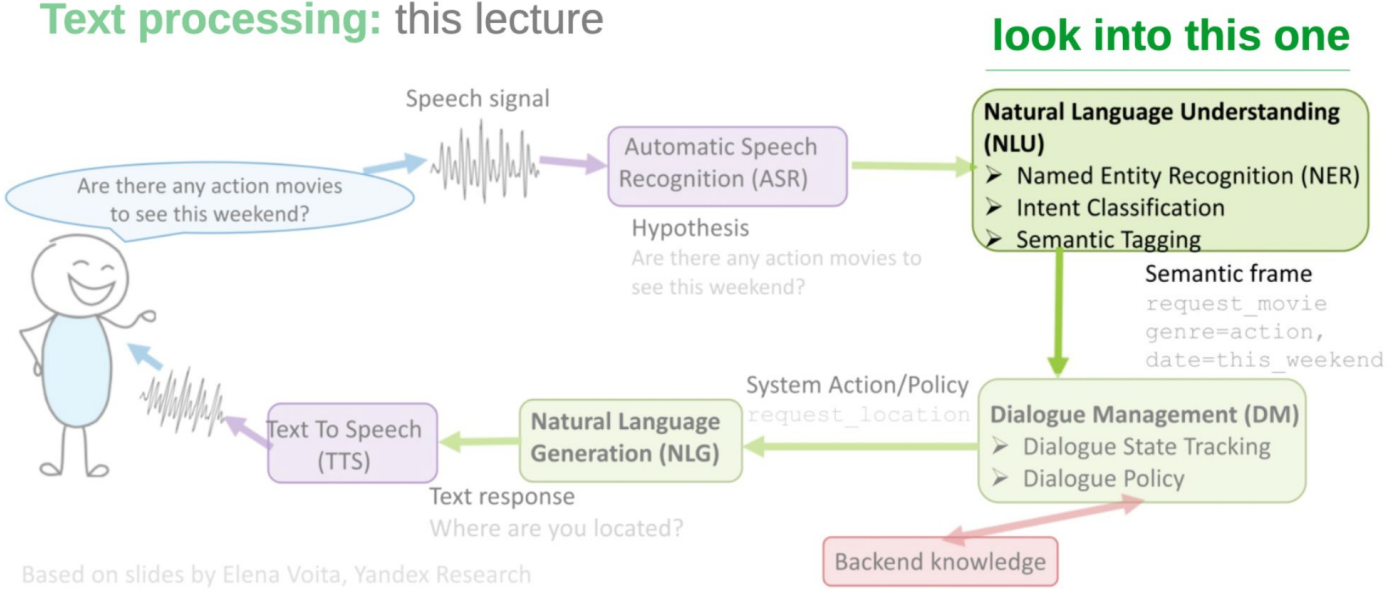


Based on slides by Elena Voita, Yandex Research

Dialogue assistants: Cascade pipeline

System design: let's split "chat bot" into smaller problems

Speech processing: see YSDA speech course
Business knowledge: rules for banking / psychology / ...
Text processing: this lecture



Based on slides by Elena Voita, Yandex Research

Dialogue assistants: Cascade pipeline

Named Entity Recognition

Why: extract keywords from user's message, use them , e.g.
- for web search, when checking a fact
- for map look-up ("where can I buy pizza in Taganrog?")
- to play music / video ("play Eminem's latest song")

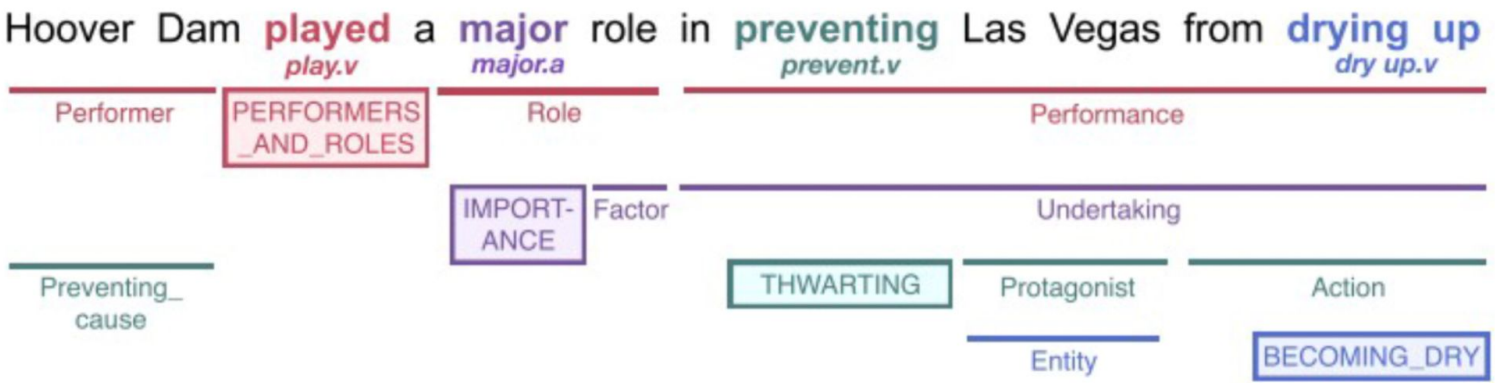
Q: how do we solve this?

When Sebastian Thrun PERSON started working on self - driving cars at Google ORG in 2007 DATE , few people outside of the company took him seriously . " I can tell you very senior CEOs of major American NORP car companies would shake my hand and turn away because I was n't worth talking to , " said Thrun PERSON , in an interview with Recode ORG earlier this week DATED .

Image credit: nanonets.com

Dialogue assistants: Cascade pipeline

- **Named Entity Recognition:** see previous slide
- **Semantic parsing:** to figure out what user wants from you

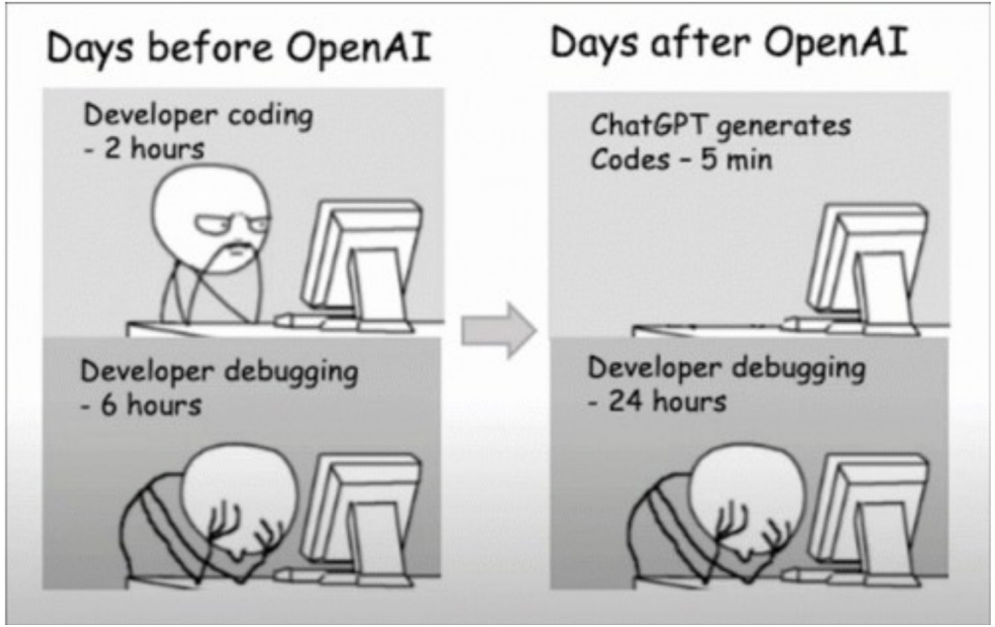


Dialogue assistants: Cascade pipeline

- **Named Entity Recognition:** see previous slide
- **Semantic parsing:** to figure out what user wants from you
- **Anaphora resolution:** to find what “it” or “this bar” refers to

Dialogue assistants: e2e system

the ChatGPT part



GPT-based assistants

GPT pretraining is not aimed at following instructions

Prompt:
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
     C = [0 for i in range(r + 1)];
     C[0] = 1;
     for i in range(1, n + 1):
          j = min(i, r);
          while j > 0:
               C[j] += C[j - 1];
               j -= 1;
     return C[r]
```

GPT-3 175B completion:
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

GPT-based assistants

GPT pretraining is not aimed at following instructions

Prompt:
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
    C = [0 for i in range(r + 1)];
    C[0] = 1;
    for i in range(1, n + 1):
        j = min(i, r);
        while j > 0:
            C[j] += C[j - 1];
            j -= 1;
    return C[r]
```

GPT-3 175B completion:
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

Solution?

Instruction Tuning

Training objective:

what we **want** vs
what we **told** model to do

What we **told** model to do:
- predict the next token on a webpage from the internet

Alignment →

What we want model to do:
- follow the user's instructions helpfully and safely

The language modeling objective is misaligned

Source: Training language models to follow instructions with human feedback, NeurIPS 2022

Instruction Tuning: InstructGPT

Step 1

Collect demonstration data, and train a supervised policy.

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.

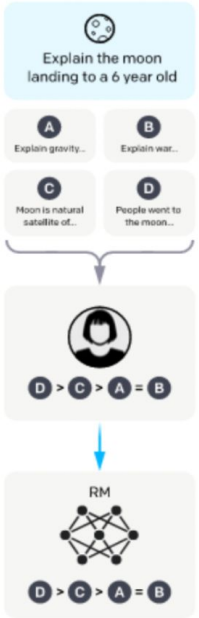
Step 2

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

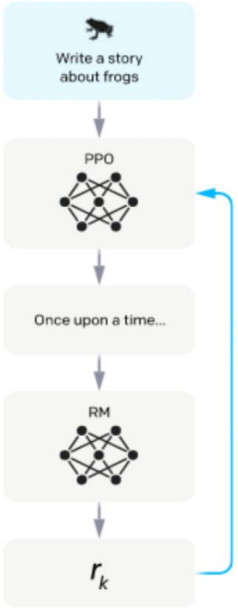
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

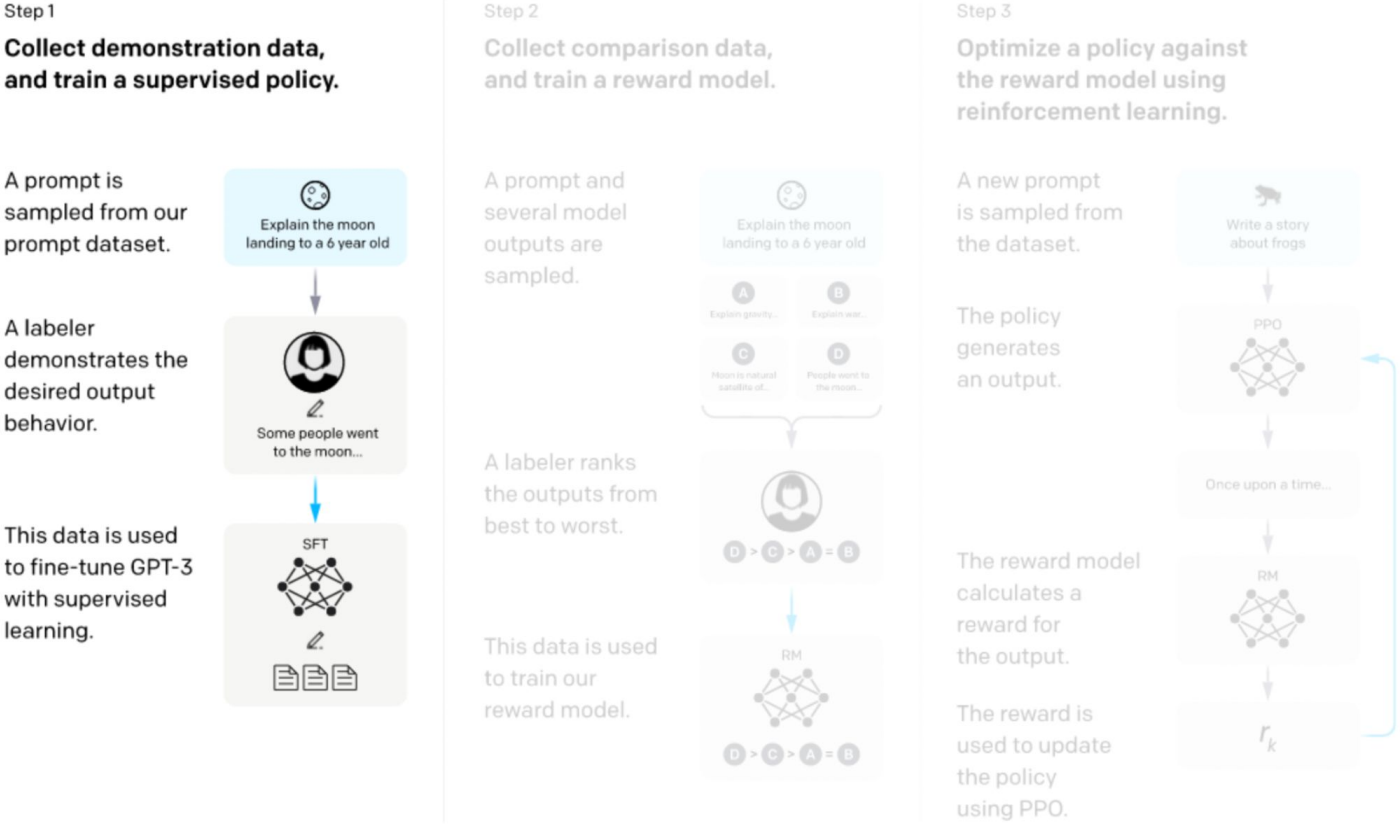
The policy generates an output.

The reward model calculates a reward for the output.

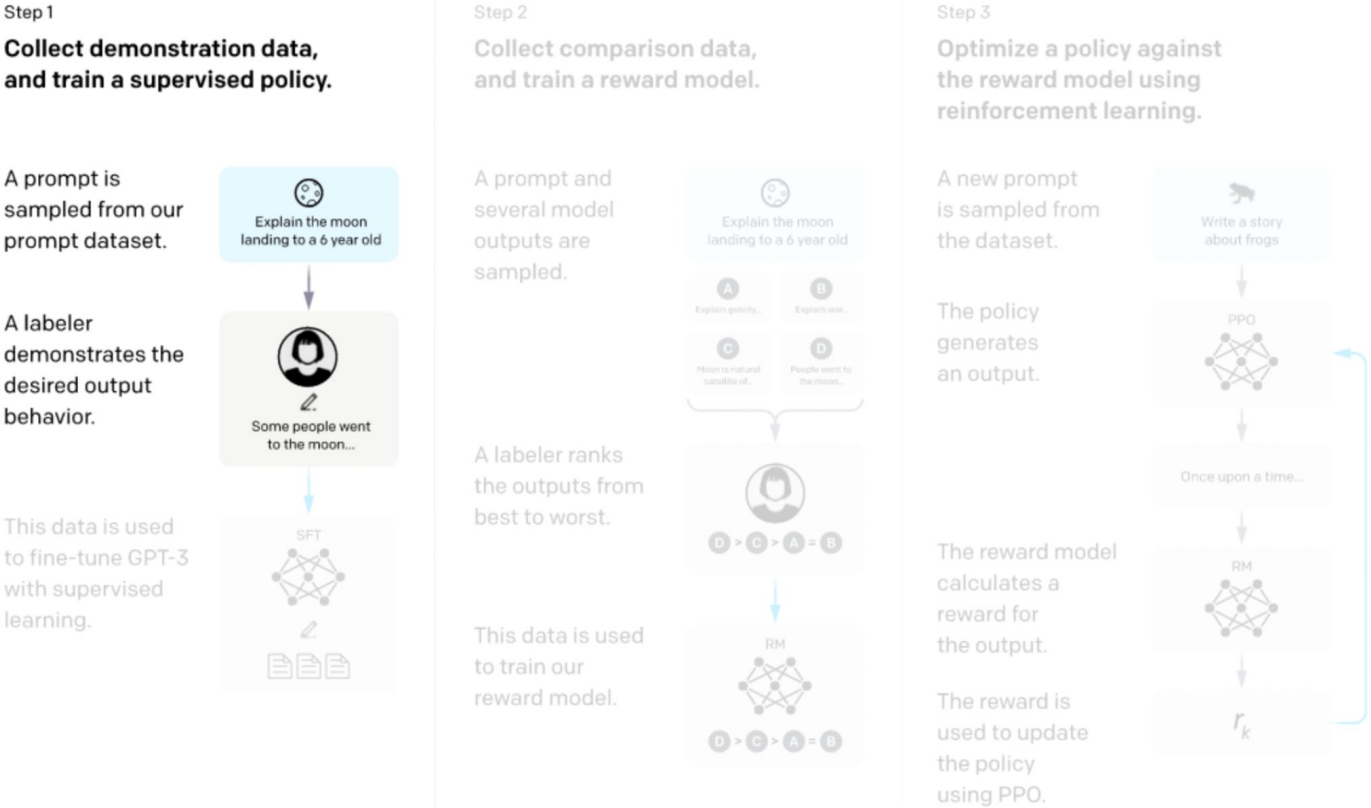
The reward is used to update the policy using PPO.



Instruction Tuning: InstructGPT



Instruction Tuning: InstructGPT



Stage 1: Supervised Finetuning

Prompt Dataset

Initial stage

Manually written by labelers

- Plain: come up with an arbitrary task, while ensuring the tasks had sufficient diversity
- Few-shot: come up with an instruction, and multiple query/response pairs for that instruction
- User-based: come up with prompts corresponding to some of the use-cases stated in waitlist applications to the OpenAI API

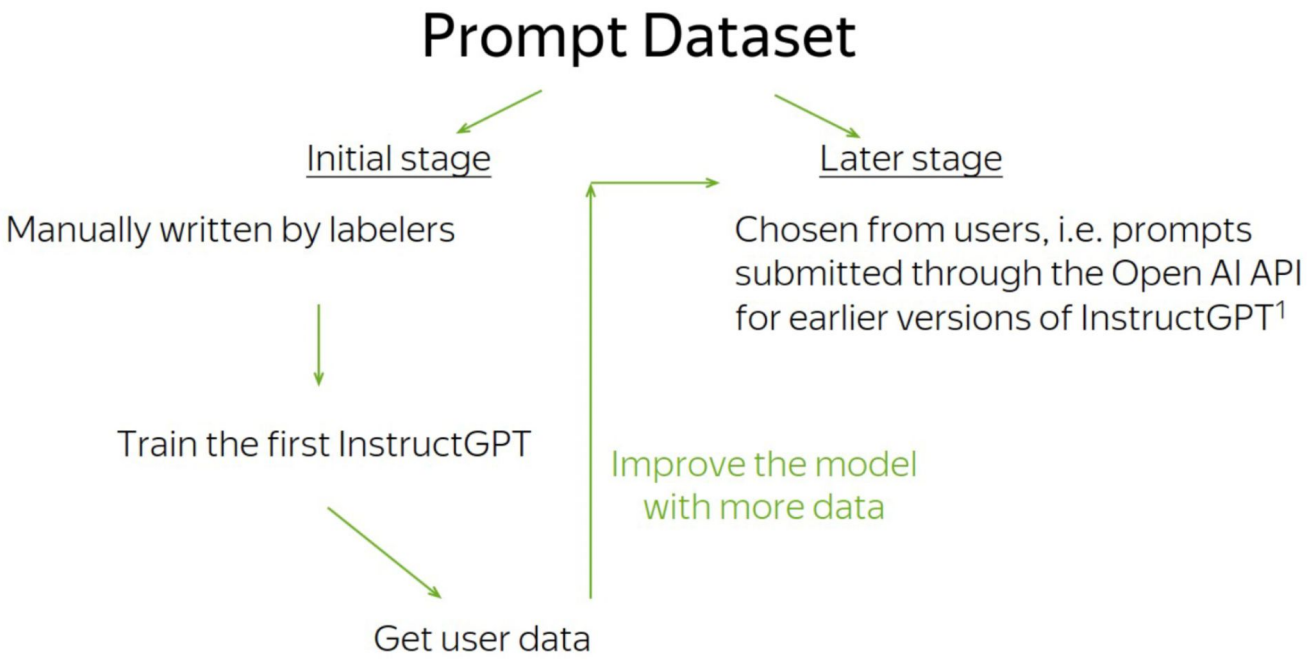
Later stage

Chosen from users, i.e. prompts submitted through the Open AI API for earlier versions of InstructGPT[1]

- filter all prompts in the training split for personally identifiable information (PII)
- heuristically deduplicate prompts
- no more than 200 prompts per user ID
- create train, validation, and test splits based on user ID

[1]https://beta. openai.com/playground

Stage 1: Supervised Finetuning



[1]https://beta. openai.com/playground

Stage 1: Supervised Finetuning

Use cases

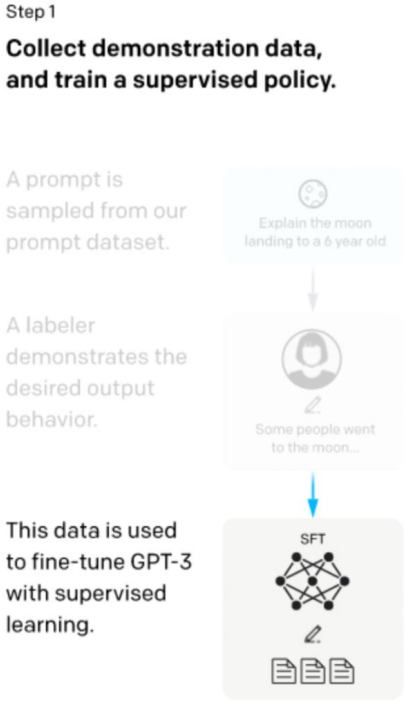
Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Examples

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.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

Source: Training language models to follow instructions with human feedback, NeurIPS 2022

Stage 1: Supervised Finetuning



Fine-tuning procedure:

exactly the same as in pre-training

minimize crossentropy with Adam

using the instruction-following data

Cheaper version: train LoRA adapters

https://arxiv.org/abs/2305.14314

InstructGPT

Why can’t we stop at SFT stage?

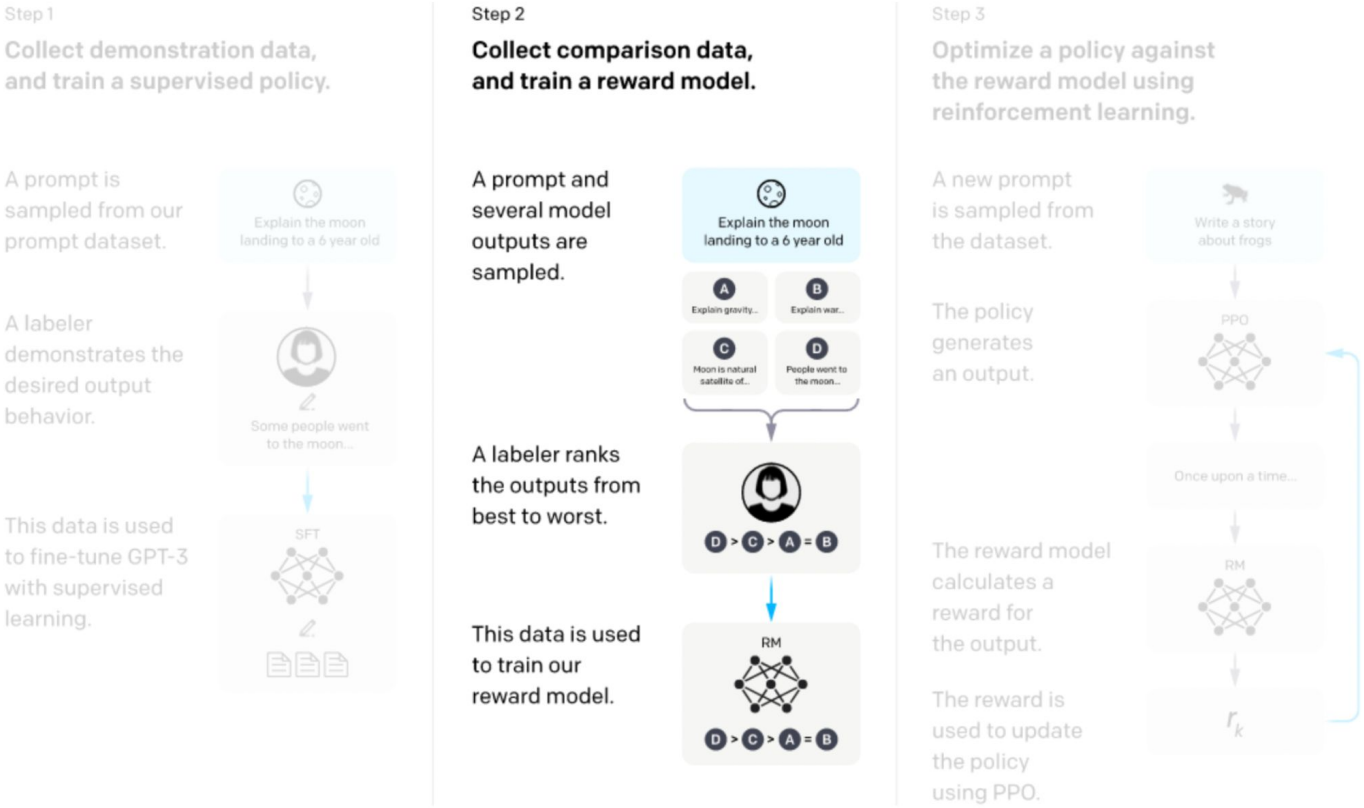
Reason 1: ranking is easier than writing for labelers
(except maybe for fact-checking)

Reason 2: supervised fine-tuning promotes hallucinations

More supervised finetuning problems

- Expensive data collection (hard to achieve answer space coverage)
- Problems with modeling
  a. Xent penalizes all errors equally
  b. Ground truth is technically one-hot -> does not estimate “answer quality” directly
- Exposure bias

Stage 2: Reward Model



Stage 2: Reward Model

How OpenAI did it

Choose:

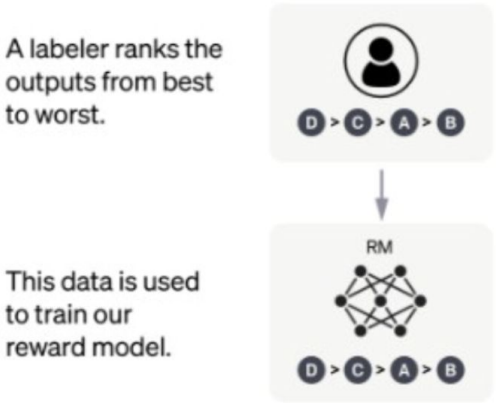
- 40 contractors on Upwork and through ScaleAI
- labelers who were sensitive to the preferences of different demographic groups
- labelers who were good at identifying outputs that were potentially harmful

Mentor:

- Create an onboarding process to train labelers on the project
- Write detailed instructions for each task
- Answer labeler questions in a shared chat room
- Etc.

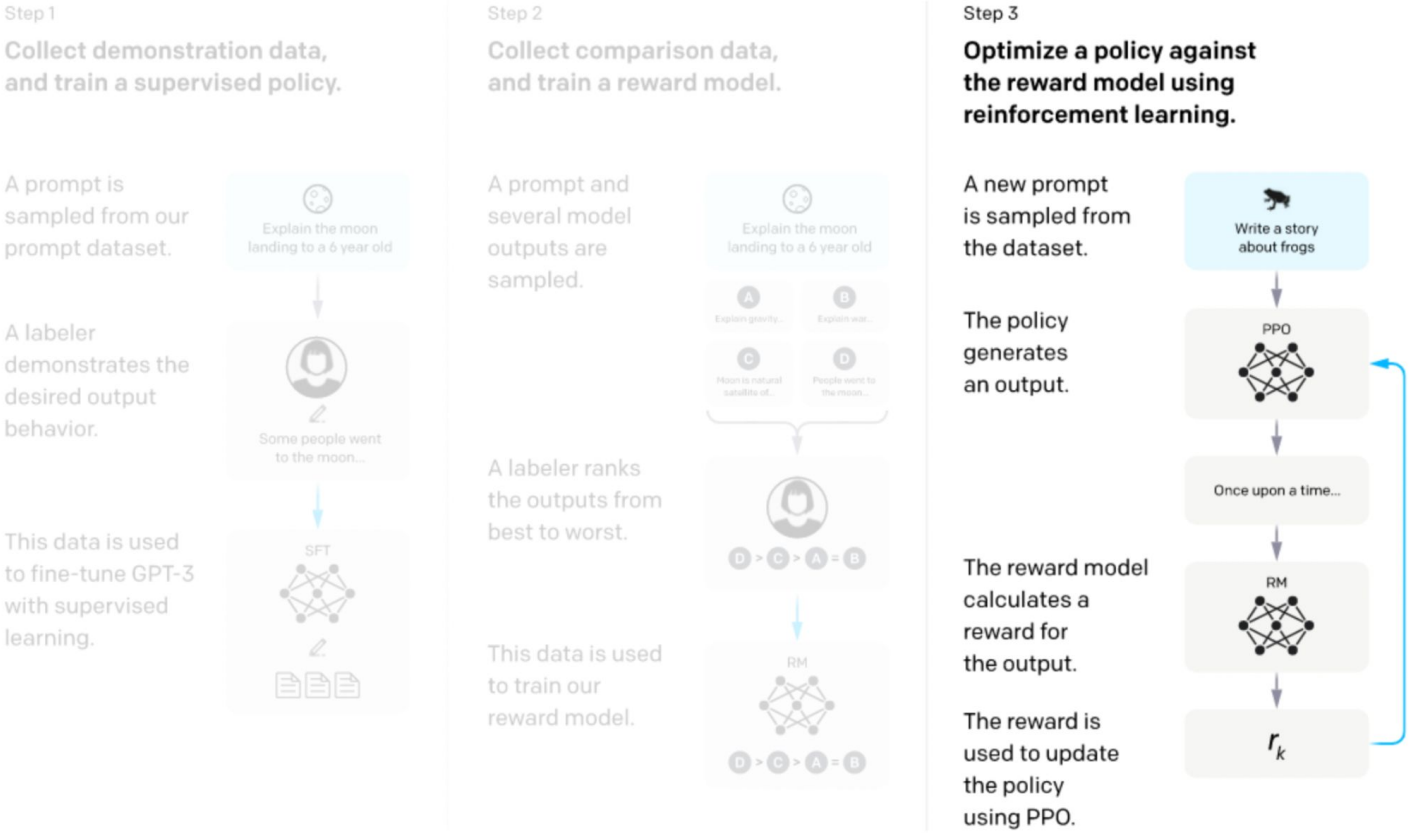
Stage 2: Reward Model

- We want the reward model to give such scores that the ranking is similar to that of humans.
- For every pair the ranking is wrong, the reward model is penalized.



$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x,y_w,y_l)\sim D}\left[\log\left(\sigma\left(r_\theta\left(x,y_w\right) - r_\theta\left(x,y_l\right)\right)\right)\right]$$

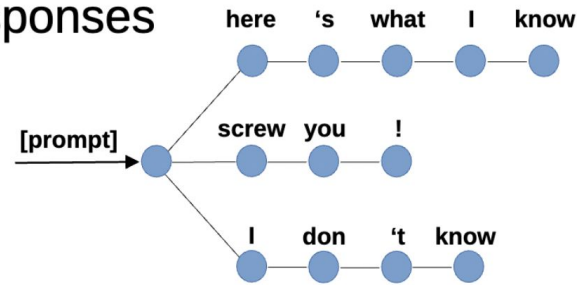
Stage 3: RL Finetuning



Self-critical sequence training

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)

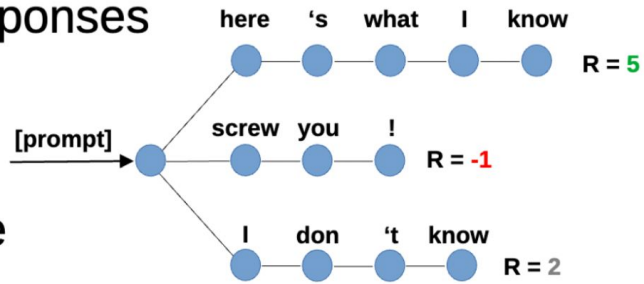


Self-critical sequence training

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)

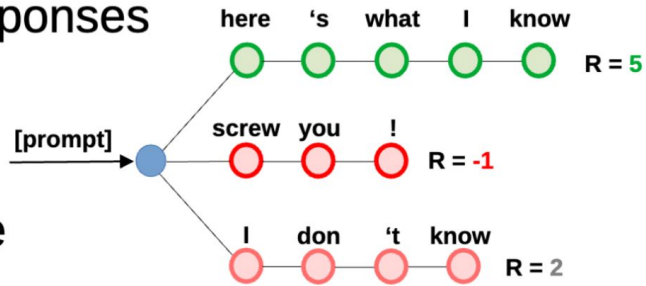
2. Compute reward for each response (apply reward model)



Self-critical sequence training

TL;DR reinforcement learning for LLM tasks:

1. Let the model generate several responses (sample with probability)



2. Compute reward for each response (apply reward model)

3. Train to increase probability of responses **with high reward** (and decrease probability if reward is low)

PPO

1. TRPO (Trust Region Policy Optimization)

Surrogate loss: $\max_{\pi} L(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a) \right]$

Constraint: $\mathbb{E}_{\pi_{\text{old}}} \left[KL(\pi || \pi_{\text{old}}) \right] \leq \epsilon$

for iteration=1, 2, . . . **do**
  Run policy for T timesteps or N trajectories
  Estimate advantage function at all timesteps
  Compute policy gradient g
  Use CG (with Hessian-vector products) to compute $F^{-1}g$
  Do line search on surrogate loss and KL constraint
end for

PPO

TRPO

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta$$

PPO v1

$$\max_\theta \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] - \beta\left(\hat{\mathbb{E}}_t\left[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] - \delta\right)$$

- Pseudocode:

  for iteration=1, 2, . . . **do**

  Run policy for T timesteps or N trajectories

  Estimate advantage function at all timesteps

  Do SGD on above objective for some number of epochs

  Do dual descent update for beta

InstructGPT

Prompt:
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
     C = [0 for i in range(r + 1)];
     C[0] = 1;
     for i in range(1, n + 1):
          j = min(i, r);
          while j > 0:
               C[j] += C[j - 1];
               j -= 1;
     return C[r]
```

GPT-3 175B completion:
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

InstructGPT 175B completion:
The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

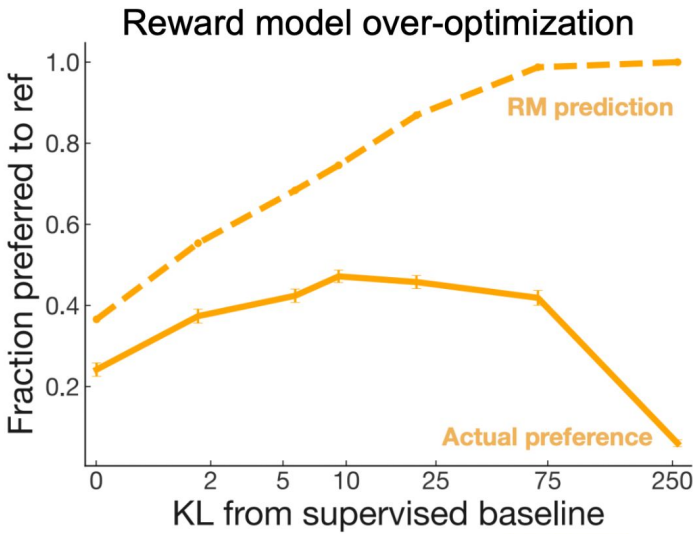
RLHF: Reward Modeling (RM)

Why do we need RM?

1. RL is sample expensive
    a. either A LOT of human evaluations needed
    b. or - train Reward Model on small corpus, train on big synthetic corpus
2. PPO if on-policy
    a. training samples for PPO need to come from distribution of current stage of model
    b. human evaluations are pre-collected -> off-policy
    c. reward model scores can be computed on policy

Problems of RLHF

- 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
- **Models** of human preferences are *even more* unreliable!

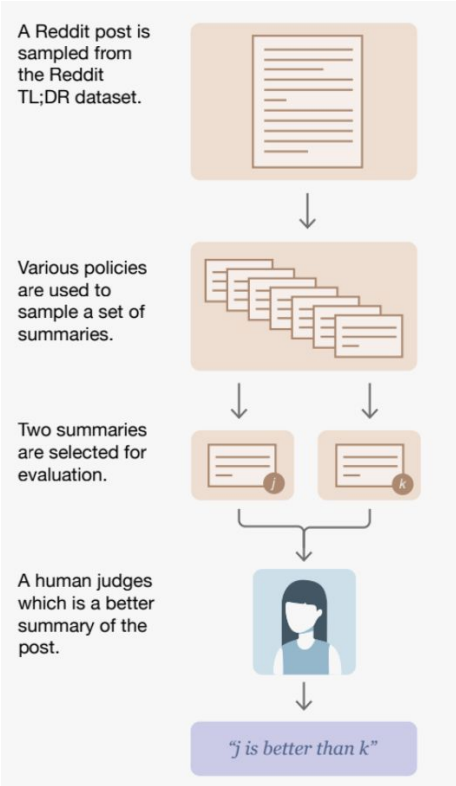


$$R(s) = RM_\phi(s) - \beta \log \left(\frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right)$$

[Stiennon et al., 2020]

Learning from Human Feedback: any alternatives?

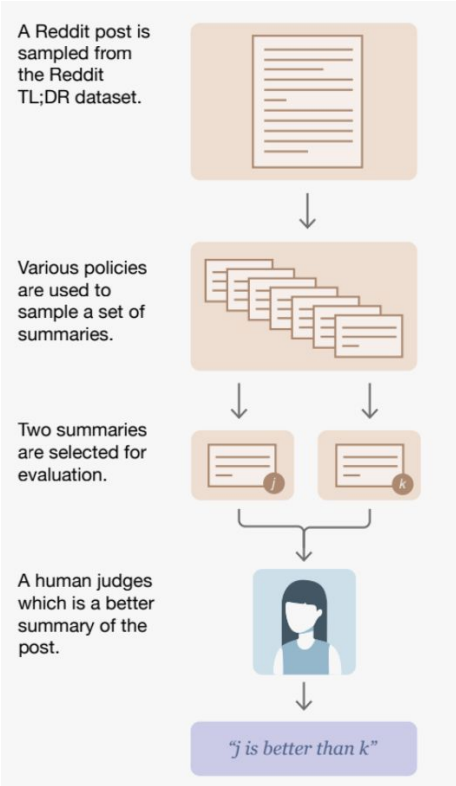
<u>Reminder:</u> main goal is to utilize answers ranking from humans



Learning from Human Feedback: any alternatives?

<u>Reminder:</u> main goal is to utilize answers ranking from humans

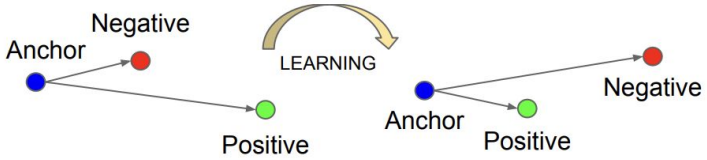
Contrastive Learning!



DSSM

DSSM (Deep Structured Semantic Model)

Textual data	→	Text encoder (word2vec, RNN etc.)	*A (anchor)*
Corresponding image	→	Image encoder (CNN etc.)	*P (positive)*
Non-correspon ding image	→	Image encoder (CNN etc.)	*N (negative)*



Triplet loss (max margin loss)

Goal: $\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$

Loss: $\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right]_+$

Triplet loss for seq2seq tasks

$$\mathcal{L}(\theta) = max(0, \delta - \log P_\theta(\mathbf{y}^+|\mathbf{x}) + \log P_\theta(\mathbf{y}^-|\mathbf{x}))$$

Triplet loss for seq2seq tasks

$$\mathcal{L}(\theta) = max(0, \delta - \log P_\theta(\mathbf{y}^+|\mathbf{x}) + \log P_\theta(\mathbf{y}^-|\mathbf{x}))$$

response with higher rating

prompt / dialogue state

response with lower rating

SLiC-HF

Contrastive Learning from Human Feedback

1. SFT model
2. Collect off-policy dataset with HF
3. **S**equence **Li**kelihood **C**alibration with **H**uman **F**eedback

SLiC-HF

Contrastive Learning from Human Feedback

1. SFT model
2. Collect off-policy dataset with HF
3. **S**equence **Li**kelihood **C**alibration with **H**uman **F**eedback

$$\mathcal{L}(\theta) = \underbrace{\sum L^{\text{cal}}(\theta, \mathbf{x}, \mathbf{y}_{\text{ref}}, \{\hat{\mathbf{y}}\}_m)}_{\textit{contrastive triplet loss}} + \underbrace{\lambda L^{\text{reg}}(\theta, \theta_{ft}; \mathbf{x}, \mathbf{y}_{\text{ref}})}_{\textit{Xent regularizer}}$$

SLiC-HF

Contrastive Learning from Human Feedback

1. SFT model
2. Collect off-policy dataset with HF
3. **S**equence **Li**kelihood **C**alibration with **H**uman **F**eedback

$$\mathcal{L}(\theta) = \underbrace{\sum L^{\text{cal}}(\theta, \mathbf{x}, \mathbf{y}_{\text{ref}}, \{\hat{\mathbf{y}}\}_m)}_{\textit{contrastive triplet loss}} + \underbrace{\lambda L^{\text{reg}}(\theta, \theta_{ft}; \mathbf{x}, \mathbf{y}_{\text{ref}})}_{\textit{Xent regularizer}}$$

$$\mathcal{L}(\theta) = max(0, \delta - \log P_\theta(\mathbf{y}^+|\mathbf{x}) + \log P_\theta(\mathbf{y}^-|\mathbf{x})) - \lambda \log P_\theta(\mathbf{y}_{\text{ref}}|\mathbf{x})$$

DPO

DPO (Direct Preference Optimization)

1. SFT model
2. Collect off-policy dataset with HF
3. DPO contrastive finetuning

DPO

DPO (Direct Preference Optimization)

1. SFT model
2. Collect off-policy dataset with HF
3. DPO contrastive finetuning

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

- Triplet max-margin loss -> logsigmoid
- Raw logP diff -> relative logP w.r.t. ref model diff

Contrastive Learning vs. RLHF

CL advantages:

- More data efficient; RL requires a lot of episodes for convergence
- More stable: PPO updates are noisy and can lead to reward hacking

CL disadvantages:

- Heavily reliant on data; can easily overfit
- If base model is of very poor quality, unstable convergence

Questions?