InstructGPT
ChatGPT

Horbach Maryna, 212

InstructGPT vs ChatGPT

- Based on GPT
- Designed for specific tasks and domains
- Reinforcement Learning from Human Feedback

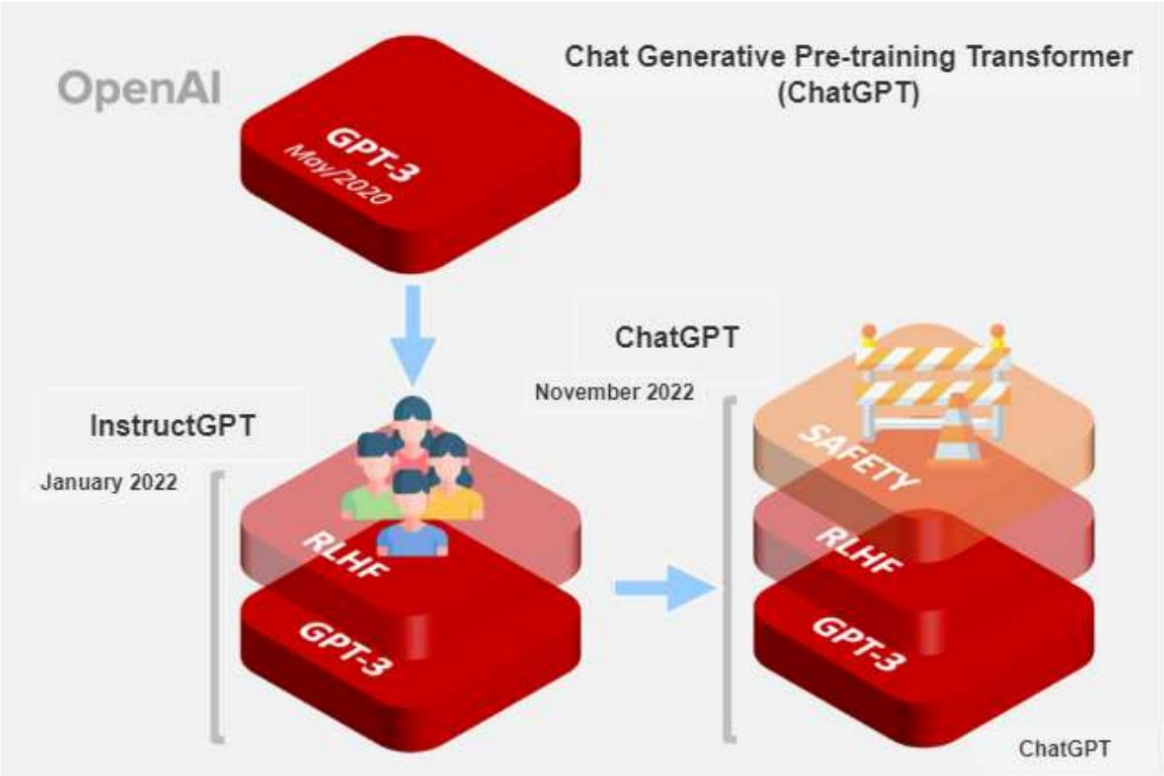
- Based on GPT
- Designed for conversational contexts
- Reinforcement Learning from Human Feedback

InstructGPT vs ChatGPT

	Instruction Following Model	Chat Completion Model
Use Cases	Task-oriented, Specific commands	Open-ended conversations, Content generation
Advantages	"Summarize this text", "Translate this sentence to French"	"Tell me a story", "What are your thoughts on XYZ?"
Examples	Precise output, Easy automation	More flexible, Handles ambiguity
Best For	Well-defined tasks, Automations	Creative or exploratory interactions

https://www.youtube.com/watch?v=C-JV0VEzn-0

InstructGPT vs ChatGPT



GPT-3 (Generative Pre-trained Transformer)

- autoregressive language model with 175 billion parameters
- model was taught to anticipate the next word in a string of text based on the words that came before it
- trained on a large collection of text data that comprised books, articles, and online content

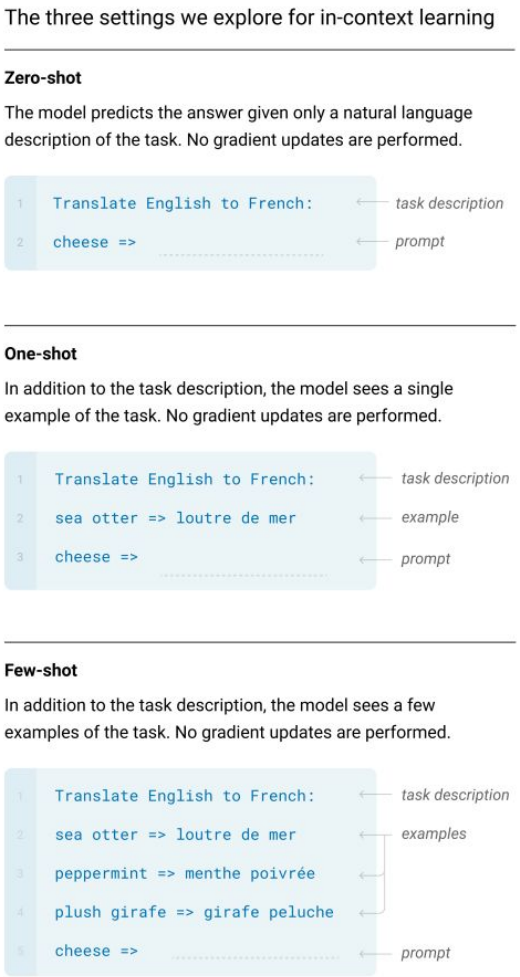
GPT-3

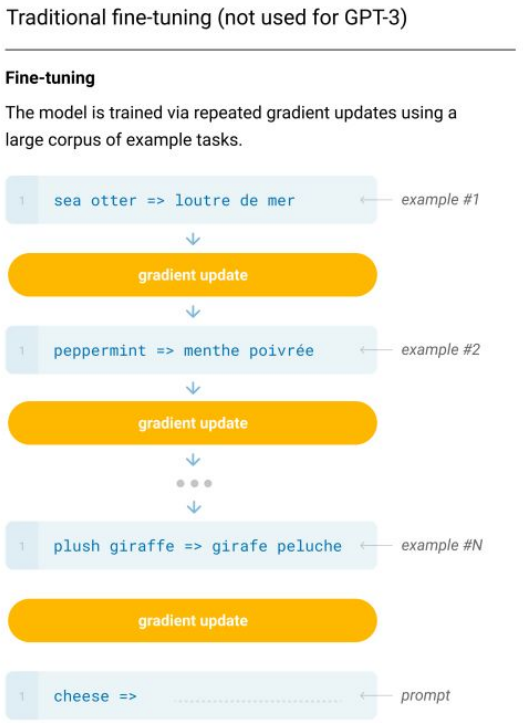
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

GPT-3

- Zero-shot
- One-shot
- Few-shot
- Fine-tuning





GPT-3

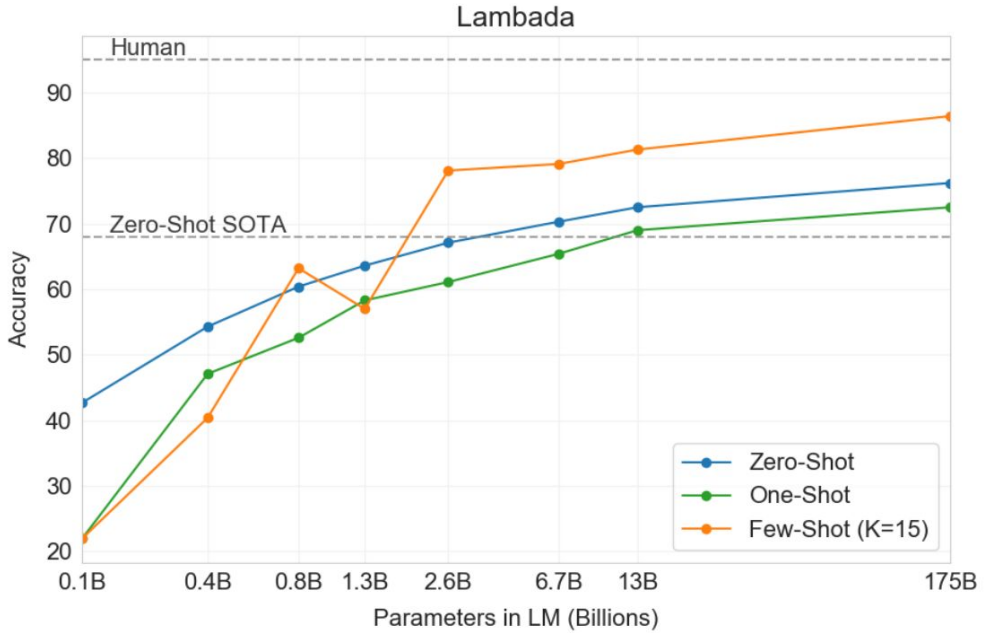


Figure 3.2: On LAMBADA, the few-shot capability of language models results in a strong boost to accuracy. GPT-3 2.7B outperforms the SOTA 17B parameter Turing-NLG [Tur20] in this setting, and GPT-3 175B advances the state of the art by 18%. Note zero-shot uses a different format from one-shot and few-shot as described in the text.

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Table 3.3: Results on three Open-Domain QA tasks. GPT-3 is shown in the few-, one-, and zero-shot settings, as compared to prior SOTA results for closed book and open domain settings. TriviaQA few-shot result is evaluated on the wiki split test server.

ChatGPT

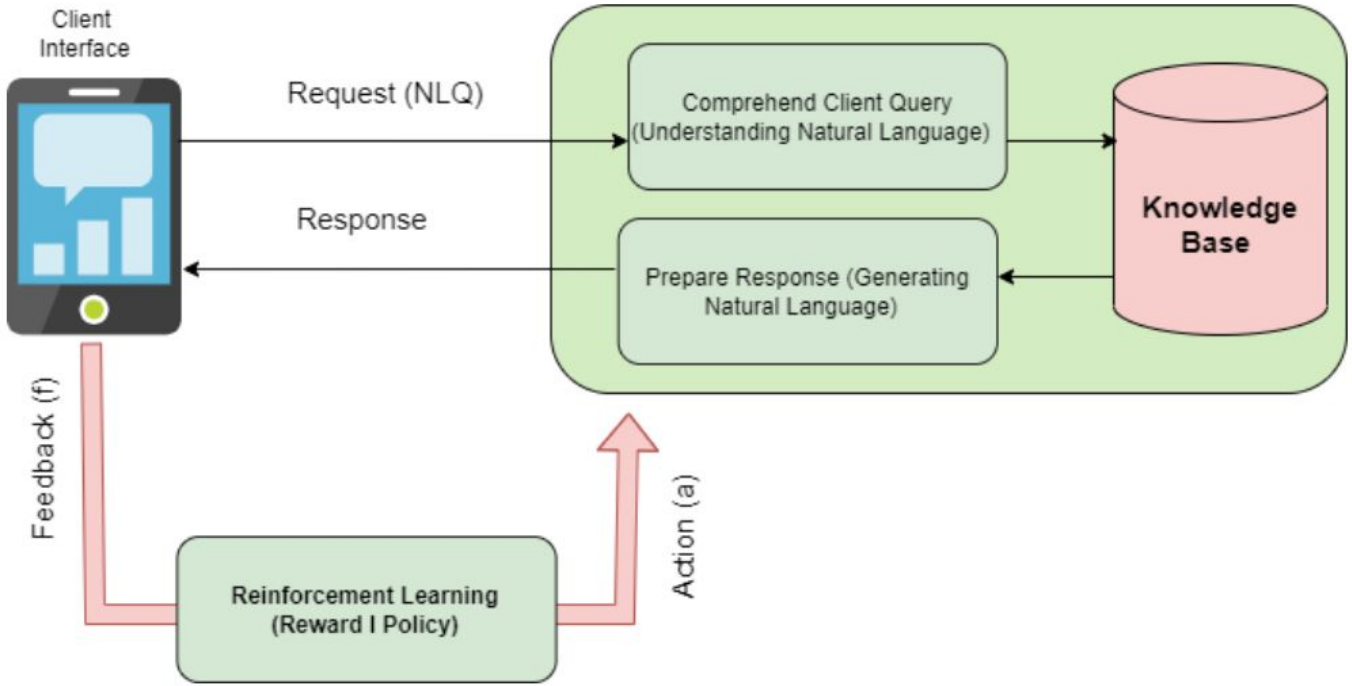
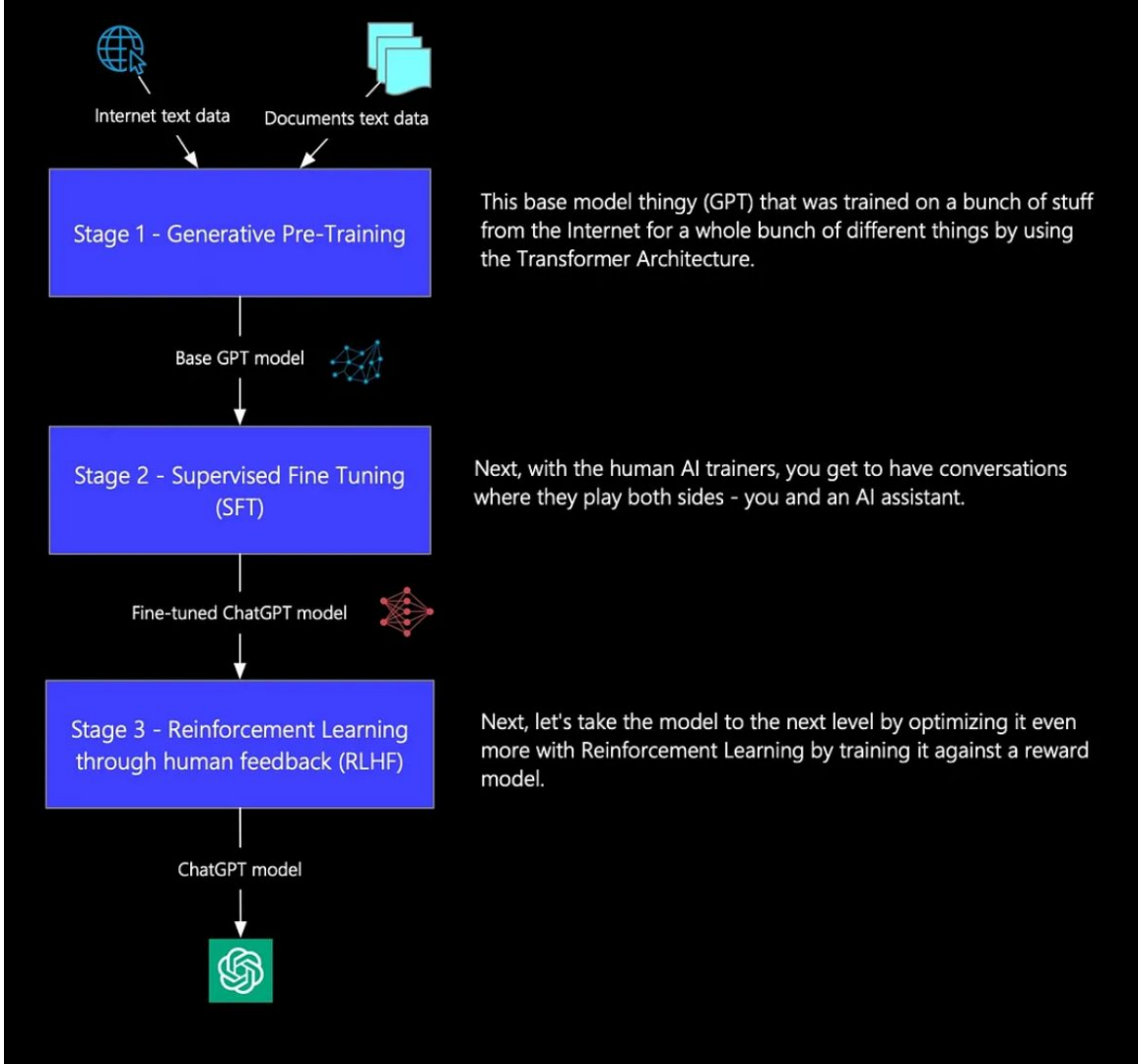


Figure 2: General Architecture of ChatGPT

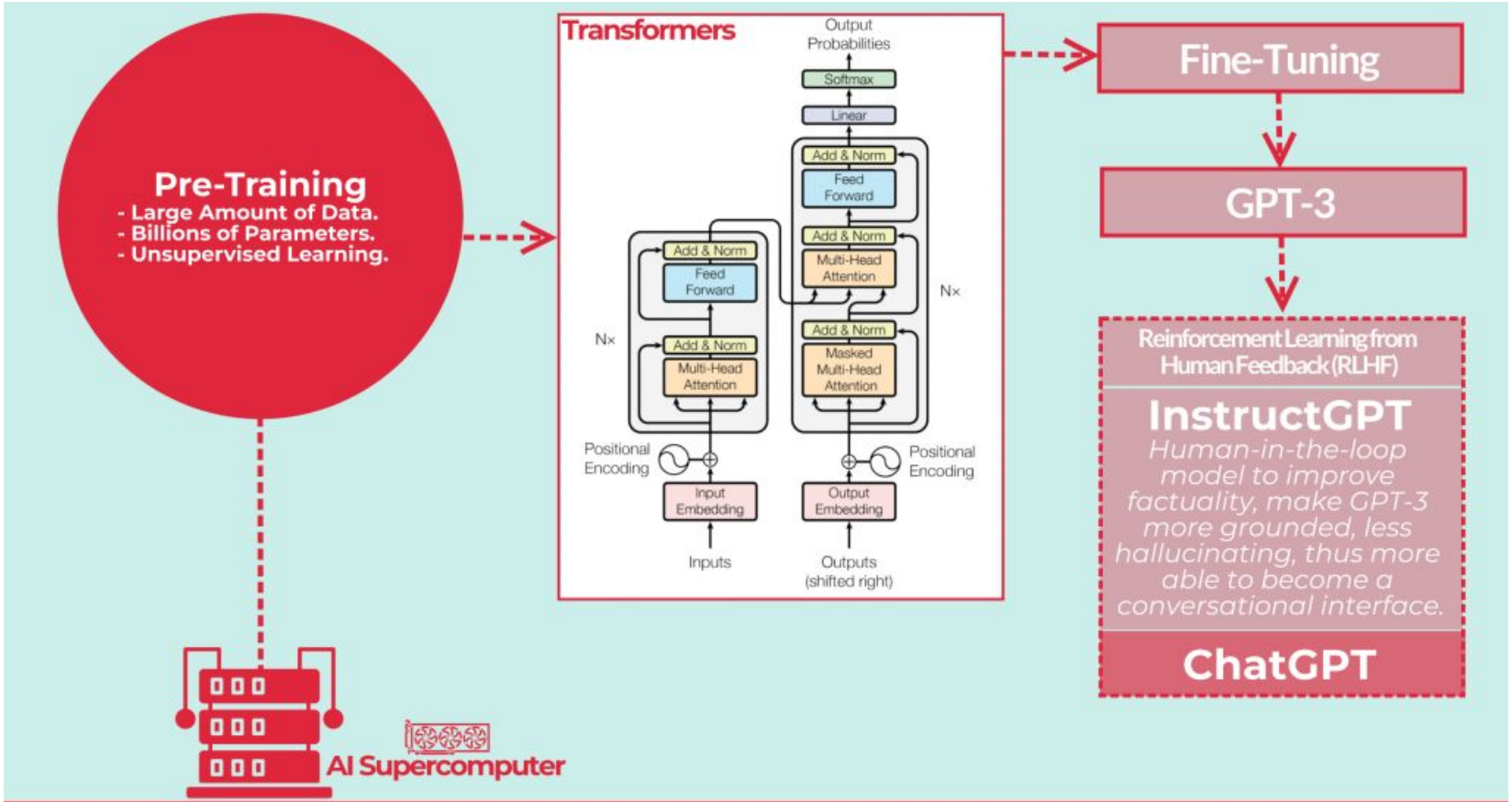
ChatGPT

SFT:

1. create sets of carefully crafted conversations
2. create the training corpus, which involves using the conversation history
3. train using SGD



InstructGPT



Problems of GPT-3:

- making up facts
- generating biased or toxic text
- not following user instructions

Solutions:

1. Supervised fine-tuning (SFT)
    - fine-tune GPT-3 on labeler demonstrations using supervised learning
    - trained for 16 epochs
    - cosine learning rate decay, residual dropout of 0.2.
2. Reward modeling (RM)
3. Reinforcement learning (RL)

Solutions: RM

RM input: prompt and response

RM output: scalar reward

Specifically, the loss function for the reward model is:

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

where $r_\theta(x, y)$ is the scalar output of the reward model for prompt x and completion y with parameters θ, y_w is the preferred completion out of the pair of y_w and y_l, and D is the dataset of human comparisons.

K - number of provided responses for each prompt

Solutions: RL

- Fine-tune the SFT model on our environment using PPO
- Environment: bandit environment

  random customer prompt — (to play arm) response to the prompt — reward (determined by RM)

- PPO-ptx: mixing the pretraining gradients into the PPO gradients, in order to fix the performance regressions on public NLP datasets

$$\begin{aligned}\text{objective}\,(\phi) = & E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]\end{aligned} \tag{2}$$

where π_ϕ^{RL} is the learned RL policy, π^{SFT} is the supervised trained model, and D_{pretrain} is the pretraining distribution. The KL reward coefficient, β, and the pretraining loss coefficient, γ, control the strength of the KL penalty and pretraining gradients respectively. For "PPO" models, γ is set to 0.

PPO: Proximal Policy Optimization

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

- alternates between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent
- whereas standard policy gradient methods perform one gradient update per data sample, it proposes a novel objective function that enables multiple epochs of minibatch updates.

Reinforcement learning from human feedback (RLHF)

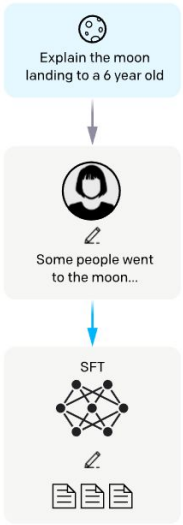
Step 1

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.



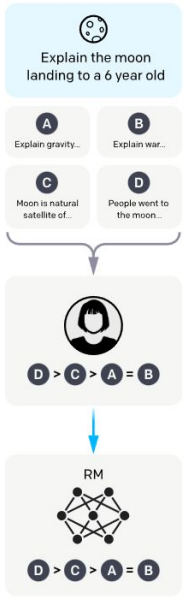
Step 2

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

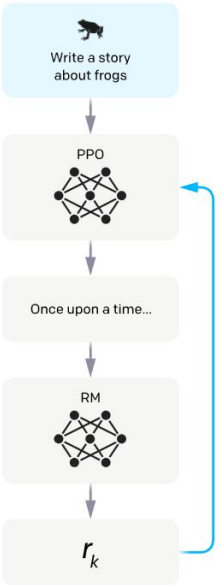
Optimize a policy against the reward model using reinforcement learning.

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.



Data for comparison

Table 1: Distribution of use case categories from our API prompt dataset.

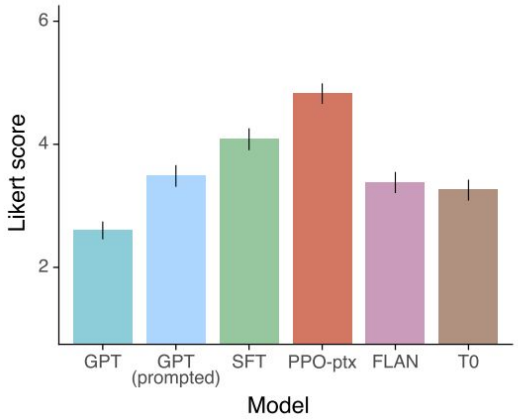
Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

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: """

Results

- Labelers significantly prefer InstructGPT outputs over outputs from GPT-3



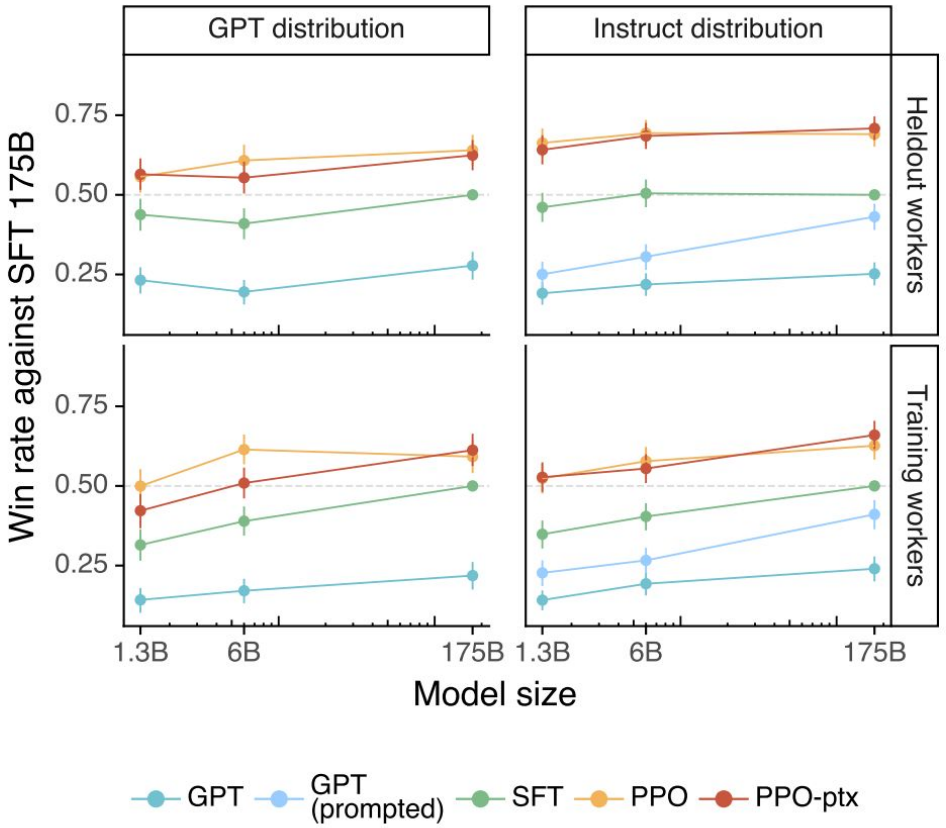


Figure 3: Preference results of our models, measured by winrate against the 175B SFT model. Left: results on prompts submitted to GPT models on the API; Right: results on prompts submitted to InstructGPT models on the API; Top: results from held-out labelers; Bottom: results from training labelers. We omit GPT (prompted) from the evals on prompts submitted to GPT-3 models (left) as these prompts are already designed to perform well for GPT-3, as opposed to prompts submitted to InstructGPT models (right).

Results

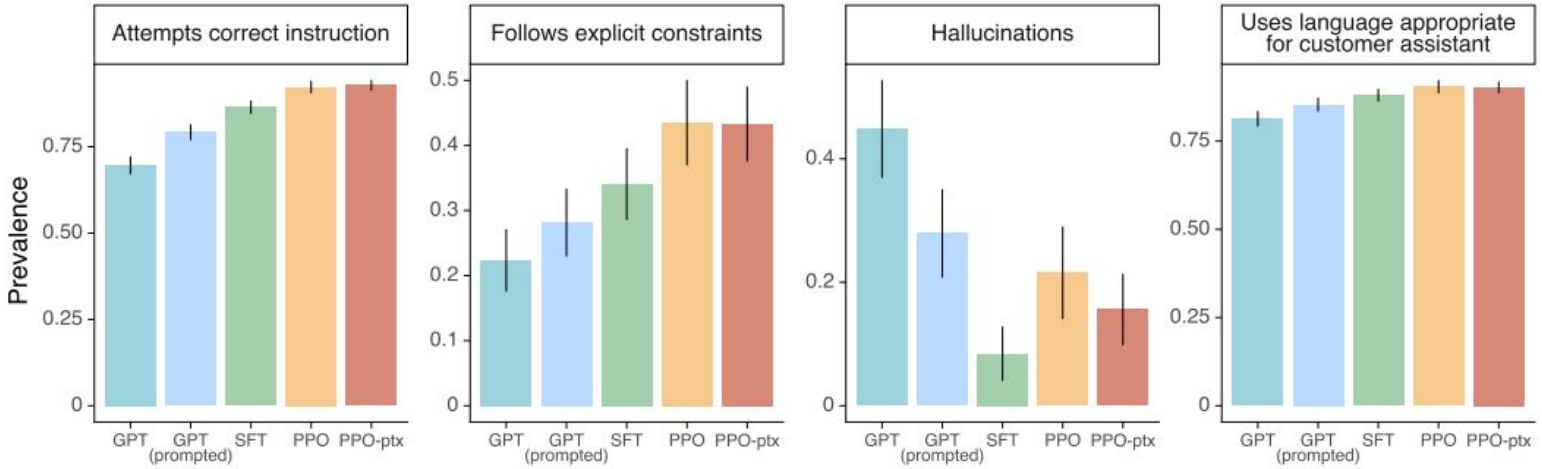
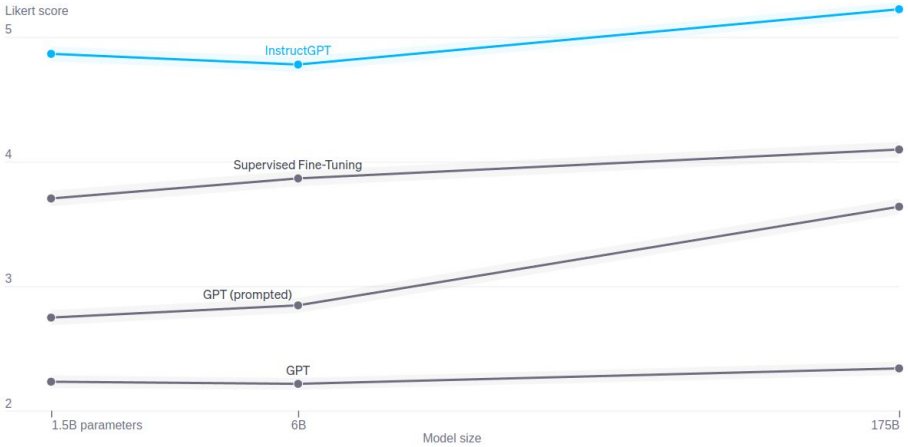


Figure 4: Metadata results on the API distribution. Note that, due to dataset sizes, these results are collapsed across model sizes. See Appendix E.2 for analysis that includes model size. Compared to GPT-3, the PPO models are more appropriate in the context of a customer assistant, are better at following explicit constraints in the instruction and attempting the correct instruction, and less likely to 'hallucinate' (meaning, making up information on closed domain tasks like summarization).

Results



Quality ratings of model outputs on a 1–7 scale (y-axis), for various model sizes (x-axis), on prompts submitted to InstructGPT models on our API. InstructGPT outputs are given much higher scores by our labelers than outputs from GPT-3 with a few-shot prompt and without, as well as models fine-tuned with supervised learning. We find similar results for prompts submitted to GPT-3 models on the API.

Dataset
RealToxicity

GPT	0.233
Supervised Fine-Tuning	0.199
InstructGPT	**0.196**

Dataset
TruthfulQA

GPT	0.224
Supervised Fine-Tuning	0.206
InstructGPT	**0.413**

API Dataset
Hallucinations

GPT	0.414
Supervised Fine-Tuning	**0.078**
InstructGPT	0.172

API Dataset
Customer Assistant Appropriate

GPT	0.811
Supervised Fine-Tuning	0.880
InstructGPT	**0.902**

Evaluating InstructGPT for toxicity, truthfulness, and appropriateness. Lower scores are better for toxicity and hallucinations, and higher scores are better for TruthfulQA and appropriateness. Hallucinations and appropriateness are measured on our API prompt distribution. Results are combined across model sizes.

Literature

1. https://openai.com/research/instruction-following
2. https://arxiv.org/pdf/2203.02155.pdf
3. https://ai.plainenglish.io/instructgpt-vs-chatgpt-fb549beba569
4. https://arxiv.org/ftp/arxiv/papers/2305/2305.15323.pdf
5. https://arxiv.org/pdf/2005.14165.pdf
6. https://rpradeepmenon.medium.com/discover-how-chatgpt-is-trained-1f20b9777d1b