A reminder

Preference Optimisation

Fine-tuning, RL, LoRA, and all that jazz

Task

- Learn about PEFT, Adaptors, LoRA, and all that jazz

- Implement an RLHF pipeline, starting from base model

    - take a base model (Qwen3)
    - do SFT on summary dataset
    - train a reward model
    - train a policy model to max rewards

- Have fun, is the last week :)

Parameter-Efficient Fine-Tuning

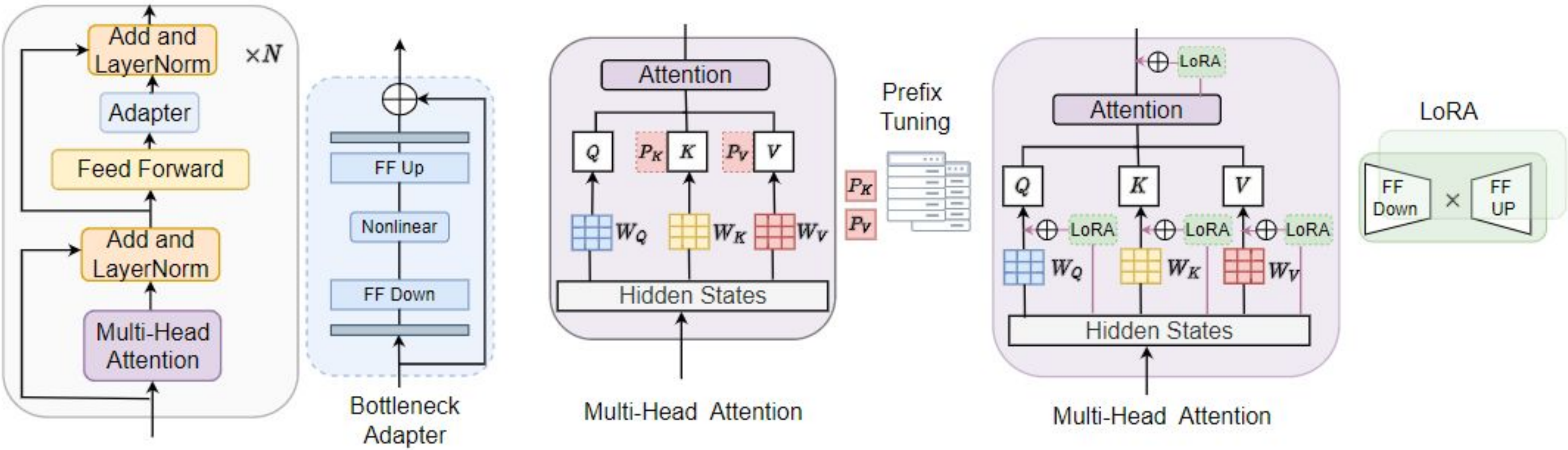
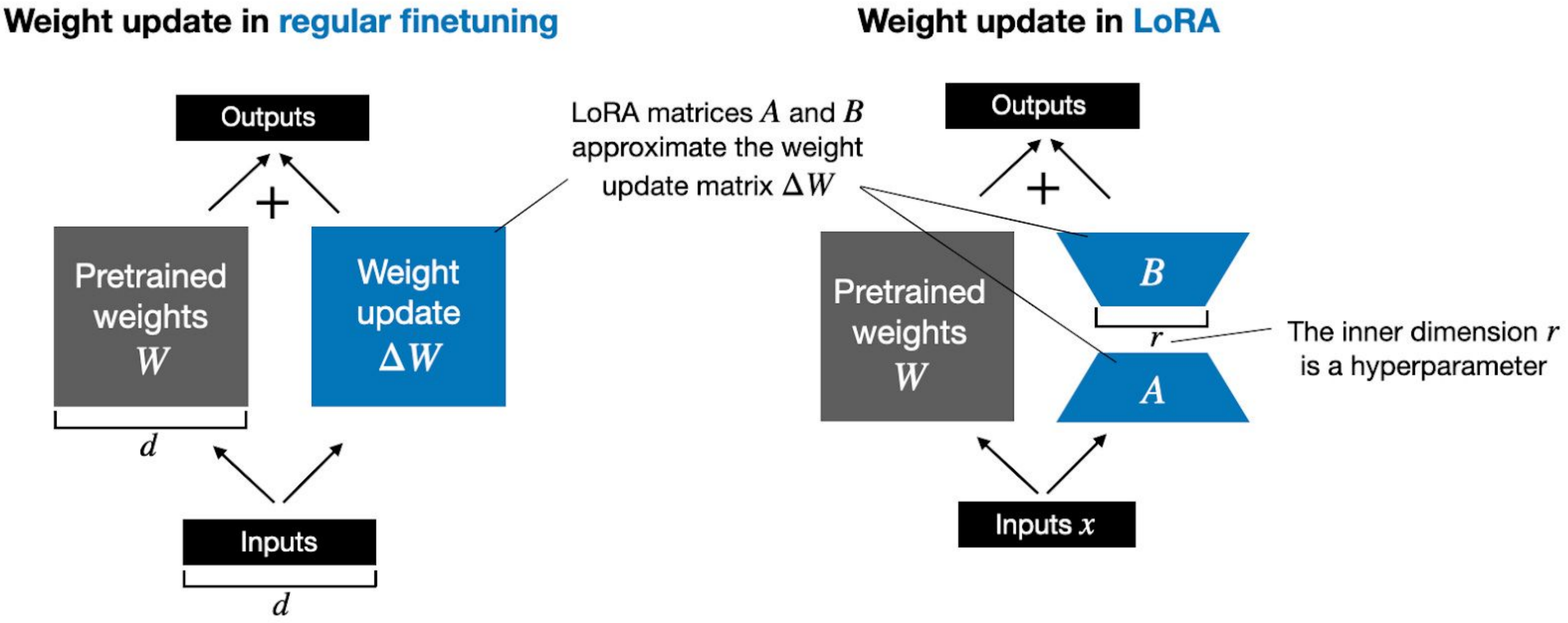


Fig. 2. Transformer architecture along with Adapter, Prefix Tuning, and LoRA.

Overview



Weight update in regular finetuning
Outputs
+
Pretrained weights W
Weight update ΔW
d
Inputs
d
LoRA matrices A and B approximate the weight update matrix ΔW
Weight update in LoRA
Outputs
+
Pretrained weights W
B
r
A
The inner dimension r is a hyperparameter
Inputs x

FOUNDERS AND CODERS

Code

Machine Learning Institute

```python
#
#
#
import torch


#
#
#
class Base(torch.nn.Module):
  def __init__(self):
    super().__init__()
    self.proj = torch.nn.Linear(10, 5)

  def forward(self, x):
    x = self.proj(x)
    return x


#
#
#
base = Base()
nums = sum(p.numel() for p in base.parameters())


#
#
#
print('Params: ', nums)
print(base)
```

```
Params:  55
Base(
  (proj): Linear(in_features=10, out_features=5, bias=True)
)
```

```python
#
#
import torch


#
#
class Base(torch.nn.Module):
  def __init__(self):
    super().__init__()
    self.proj = torch.nn.Linear(10, 5)
    self.lora = torch.nn.Sequential(
      torch.nn.Linear(10, 2, bias=False),
      torch.nn.Dropout(0.1),
      torch.nn.Linear(2, 5, bias=False),
    )

  def forward(self, x):
    x = self.proj(x)
    return x + self.lora(x)

#
#
base = Base()
nums = sum(p.numel() for p in base.parameters())


#
#
print('Params: ', nums)
print(base)
```

```
Params:  85
Base(
  (proj): Linear(in_features=10, out_features=5, bias=True)
  (lora): Sequential(
    (0): Linear(in_features=10, out_features=2, bias=False)
    (1): Dropout(p=0.1, inplace=False)
    (2): Linear(in_features=2, out_features=5, bias=False)
  )
)
```

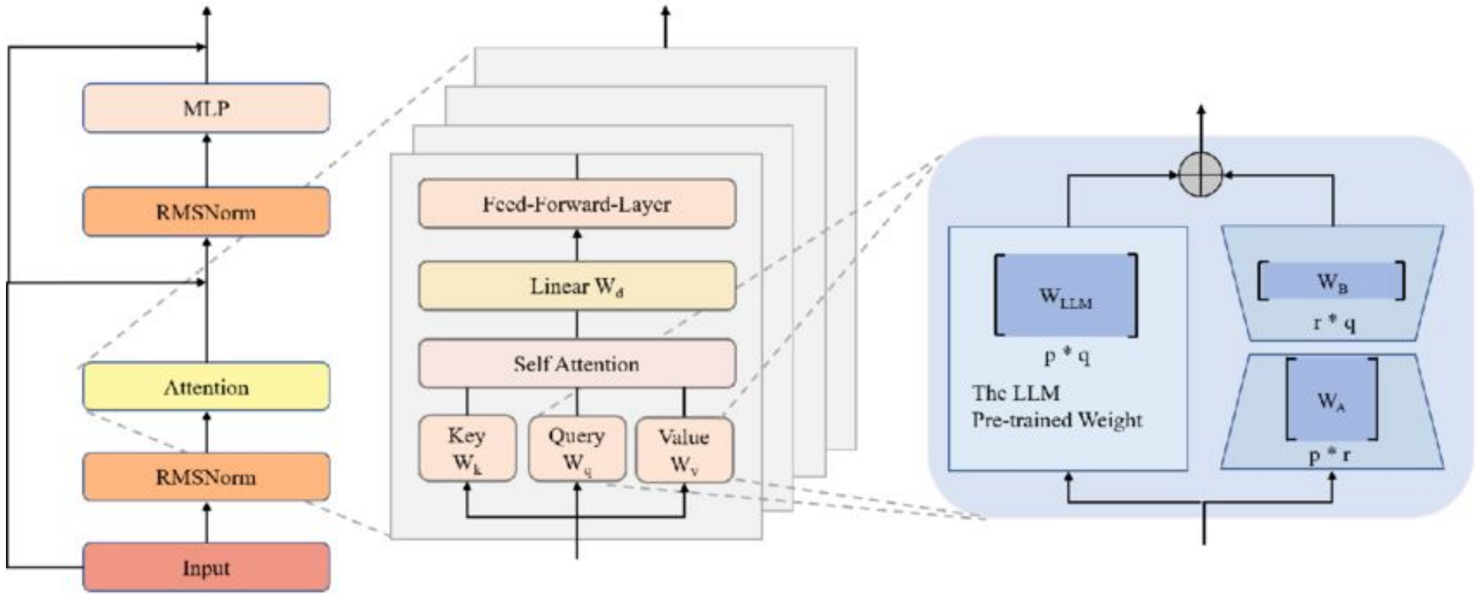
```python
#
#
import torch

#
#
class Base(torch.nn.Module):
  def __init__(self):
    super().__init__()
    self.proj = torch.nn.Linear(10, 5)
    self.adpt = torch.nn.Sequential(
      torch.nn.Linear(10, 2, bias=False),
      torch.nn.ReLU(),
      torch.nn.Dropout(0.1),
      torch.nn.Linear(2, 5, bias=False),
    )

  def forward(self, x):
    x = self.proj(x)
    return x + self.adpt(x)

#
#
base = Base()
nums = sum(p.numel() for p in base.parameters())


#
#
print('Params: ', nums)
print(base)
```

```
Params:  85
Base(
  (proj): Linear(in_features=10, out_features=5, bias=True)
  (adpt): Sequential(
    (0): Linear(in_features=10, out_features=2, bias=False)
    (1): ReLU()
    (2): Dropout(p=0.1, inplace=False)
    (3): Linear(in_features=2, out_features=5, bias=False)
  )
)
```

```python
#
#
import torch
import peft

#
#
class Base(torch.nn.Module):
  def __init__(self):
    super().__init__()
    self.proj = torch.nn.Linear(10, 5)

  def forward(self, x):
    x = self.proj(x)
    return x


#
#
conf = peft.LoraConfig(r=2, target_modules=['proj'])

#
#
base = peft.get_peft_model(Base(), conf)
nums = sum(p.numel() for p in base.parameters())


#
#
print('Params: ', nums)
print(base)
```

```
Params:  85
PeftModel(
  (base_model): LoraModel(
    (model): Base(
      (proj): lora.Linear(
        (base_layer): Linear(in_features=10, out_features=5, bias=True)
        (lora_dropout): ModuleDict(
          (default): Identity()
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=10, out_features=2, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=2, out_features=5, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
        (lora_magnitude_vector): ModuleDict()
      )
    )
  )
)
```

LoRA


MLP
RMSNorm
Attention
RMSNorm
Input
Feed-Forward-Layer
Linear W_d
Self Attention
Key W_k
Query W_q
Value W_v
W_{LLM}
p * q
The LLM Pre-trained Weight
W_B
r * q
W_A
p * r

Prefix Tuning

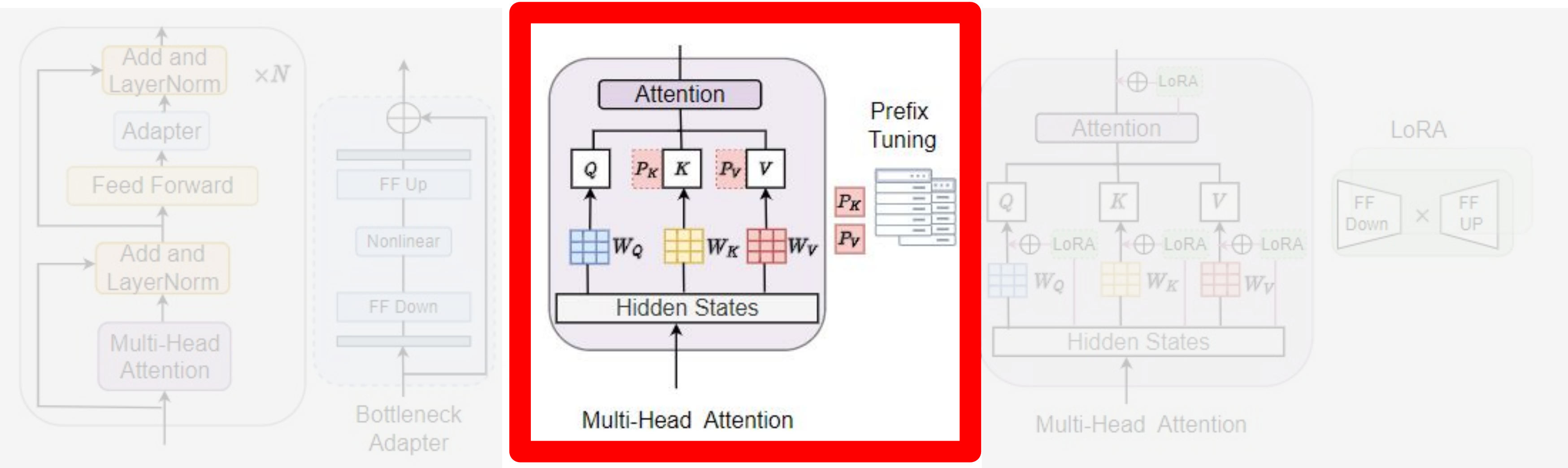


Fig. 2. Transformer architecture along with Adapter, Prefix Tuning, and LoRA.

FOUNDERS AND CODERS

Reinforcement Learning

Machine Learning Institute

arXiv:1909.08593v2 [cs.CL] 8 Jan 2020

Fine-Tuning Language Models from Human Preferences

Daniel M. Ziegler* Nisan Stiennon* Jeffrey Wu Tom B. Brown
Alec Radford Dario Amodei Paul Christiano Geoffrey Irving
OpenAI
{dmz,nisan,jeffwu,tom,alec,damodei,paul,irving}@openai.com

Abstract

Reward learning enables the application of reinforcement learning (RL) to tasks where reward is defined by human judgment, building a model of reward by asking humans questions. Most work on reward learning has used simulated environments, but complex information about values is often expressed in natural language, and we believe reward learning for language is a key to making RL practical and safe for real-world tasks. In this paper, we build on advances in generative pretraining of language models to apply reward learning to four natural language tasks: continuing text with positive sentiment or physically descriptive language, and summarization tasks on the TL;DR and CNN/Daily Mail datasets. For stylistic continuation we achieve good results with only 5,000 comparisons evaluated by humans. For summarization, models trained with 60,000 comparisons copy whole sentences from the input but skip irrelevant preamble; this leads to reasonable ROUGE scores and very good performance according to our human labelers, but may be exploiting the fact that labelers rely on simple heuristics.

1. Introduction

We would like to apply reinforcement learning to complex tasks defined only by human judgment, where we can only tell whether a result is good or bad by asking humans. To do this, we can first use human labels to train a model of reward, and then optimize that model. While there is a long history of work learning such models from humans through interaction, this work has only recently been applied to modern deep learning, and even then has only been applied to relatively simple simulated environments (Christiano et al., 2017; Ibarz et al., 2018; Bahdanau et al., 2018). By contrast, real world settings in which humans need to specify complex goals to AI agents are likely to both involve and require natural language, which is a rich medium for expressing value-laden concepts. Natural language is particularly important when an agent must communicate back to a human to help provide a more accurate supervisory signal (Irving et al., 2018; Christiano et al., 2018; Leike et al., 2018).

Natural language processing has seen substantial recent advances. One successful method has been to pretrain a large generative language model on a corpus of unsupervised data, then fine-tune the model for supervised NLP tasks (Dai and Le, 2015; Peters et al., 2018; Radford et al., 2018; Khandelwal et al., 2019). This method often substantially outperforms training on the supervised datasets from scratch, and a single pretrained language model often can be fine-tuned for state of the art performance on many different supervised datasets (Howard and Ruder, 2018). In some cases, fine-tuning is not required: Radford et al. (2019) find that generatively trained models show reasonable performance on NLP tasks with no additional training (zero-shot).

There is a long literature applying reinforcement learning to natural language tasks. Much of this work uses algorithmically defined reward functions such as BLEU for translation (Ranzato et al., 2015; Wu et al., 2016), ROUGE for summarization (Ranzato et al., 2015; Paulus et al., 2017; Wu and Hu, 2018; Gao et al., 2019b), music theory-based rewards (Jaques et al., 2017), or event detectors for story generation (Tambwekar et al., 2018). Nguyen et al. (2017) used RL on BLEU but applied several error models to approximate human behavior. Wu and Hu (2018) and Cho et al. (2019) learned models of coherence from existing text and used them as RL rewards for summarization and long-form generation, respectively. Gao et al. (2019a) built an interactive summarization tool by applying reward learning to one article at a time. Experiments using human evaluations as rewards include Kreutzer et al. (2018) which used off-policy reward learning for translation, and Jaques et al. (2019) which applied the modified Q-learning methods of Jaques et al. (2017) to implicit human preferences in dialog. Yi et al. (2019) learned rewards from humans to fine-tune dialog models, but smoothed the rewards to allow supervised learning. We refer to Luketina et al. (2019) for a survey of

*Equal contribution. Correspondence to paul@openai.com.

2019

arXiv:2009.01325v3 [cs.CL] 15 Feb 2022

Learning to summarize from human feedback

Nisan Stiennon* Long Ouyang* Jeff Wu* Daniel M. Ziegler* Ryan Lowe*
Chelsea Voss* Alec Radford Dario Amodei Paul Christiano*
OpenAI

Abstract

As language models become more powerful, training and evaluation are increasingly bottlenecked by the data and metrics used for a particular task. For example, summarization models are often trained to predict human reference summaries and evaluated using ROUGE, but both of these metrics are rough proxies for what we really care about—summary quality. In this work, we show that it is possible to significantly improve summary quality by training a model to optimize for human preferences. We collect a large, high-quality dataset of human comparisons between summaries, train a model to predict the human-preferred summary, and use that model as a reward function to fine-tune a summarization policy using reinforcement learning. We apply our method to a version of the TL;DR dataset of Reddit posts [63] and find that our models significantly outperform both human reference summaries and much larger models fine-tuned with supervised learning alone. Our models also transfer to CNN/DM news articles [22], producing summaries nearly as good as the human reference without any news-specific fine-tuning.[2] We conduct extensive analyses to understand our human feedback dataset and fine-tuned models.[3] We establish that our reward model generalizes to new datasets, and that optimizing our reward model results in better summaries than optimizing ROUGE according to humans. We hope the evidence from our paper motivates machine learning researchers to pay closer attention to how their training loss affects the model behavior they actually want.

1 Introduction

Large-scale language model pretraining has become increasingly prevalent for achieving high performance on a variety of natural language processing (NLP) tasks. When applying these models to a specific task, they are usually fine-tuned using supervised learning, often to maximize the log probability of a set of human demonstrations.

While this strategy has led to markedly improved performance, there is still a misalignment between this fine-tuning objective—maximizing the likelihood of human-written text—and what we care about—generating high-quality outputs as determined by humans. This misalignment has several causes: the maximum likelihood objective has no distinction between important errors (e.g. making up facts [41]) and unimportant errors (e.g. selecting the precise word from a set of synonyms); models

*This was a joint project of the OpenAI Reflection team. Author order was randomized amongst {LO, JW, DZ, NS}; CV and RL were full-time contributors for most of the duration. PC is the team lead.
[2]Samples from all of our models can be viewed on our website.
[3]We provide inference code for our 1.3B models and baselines, as well as a model card and our human feedback dataset with over 64k summary comparisons, here.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2020

arXiv:2203.02155v1 [cs.CL] 4 Mar 2022

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

Abstract

Making language models bigger does not inherently make them better at following a user's intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through the OpenAI API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

1 Introduction

Large language models (LMs) can be "prompted" to perform a range of natural language processing (NLP) tasks, given some examples of the task as input. However, these models often express unintended behaviors such as making up facts, generating biased or toxic text, or simply not following user instructions (Bender et al., 2021; Bommasani et al., 2021; Kenton et al., 2021; Weidinger et al., 2021; Tamkin et al., 2021; Gehman et al., 2020). This is because the language modeling objective

*Primary authors. This was a joint project of the OpenAI Alignment team. RL and JL are the team leads. Corresponding author: lowe@openai.com.
†Work done while at OpenAI. Current affiliations: AA: Anthropic; PC: Alignment Research Center.

2022

FOUNDERS AND CODERS

RLHF

Machine Learning Institute

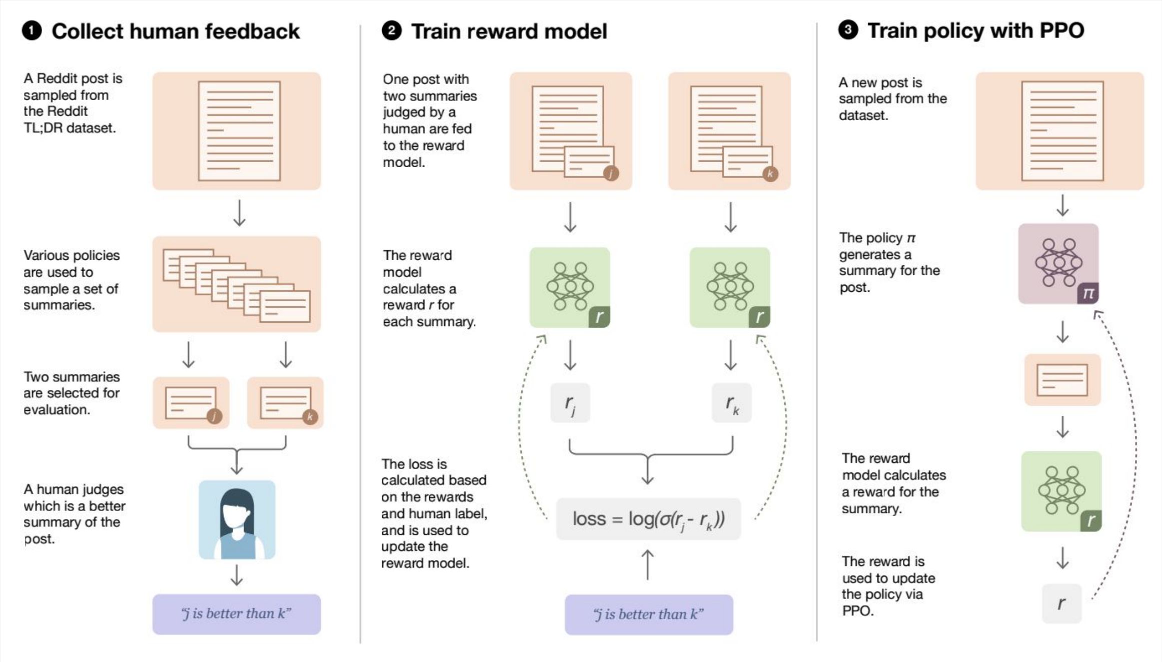
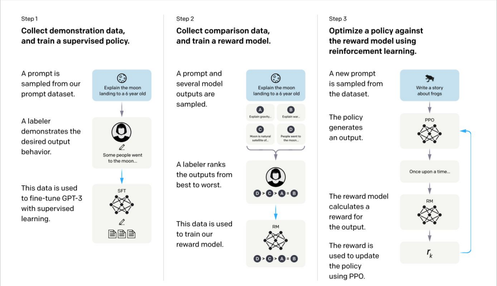


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

2020



2022

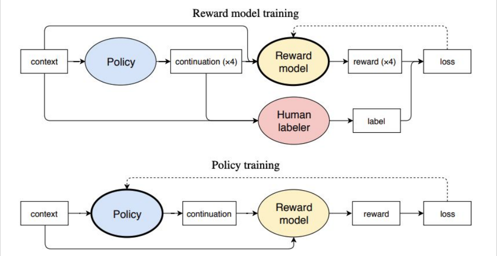


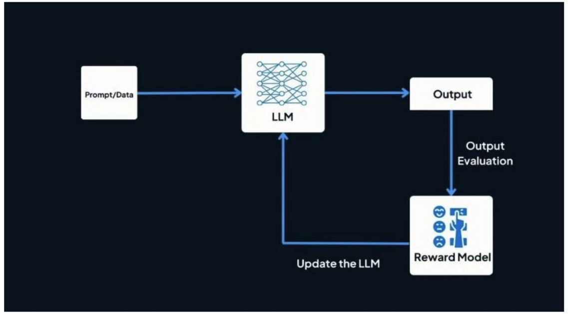
Figure 1: Our training processes for reward model and policy. In the online case, the processes are interleaved.

2019

Help

RLHF on a Budget: GPT-2 for Summarization

Star Chen · Follow · 11 min read · Apr 7, 2025



While reading *Implementing RLHF: Learning to Summarize with trlX*, which uses GPT-J to train a reinforcement learning from human feedback (RLHF) model, I started wondering: can GPT-2 hold its own against GPT-J, but with far less compute? There's something compelling about these "old" and "micro" language models — especially how fast they are to train. But can they still produce solid results?

Qwen/Qwen3-14B-Base
Text Generation • 15B • Updated May 21 • 32.5k • 27

Qwen/Qwen3-8B-Base
Text Generation • 8B • Updated May 21 • 3.34M • • 39

Qwen/Qwen3-4B-Base
Text Generation • 4B • Updated May 21 • 405k • • 34

Qwen/Qwen3-1.7B-Base
Text Generation • 2B • Updated May 21 • 115k • 24

Qwen/Qwen3-0.6B-Base
Text Generation • 0.6B • Updated May 21 • 76.3k • • 73

Good luck!

FOUNDERS AND CODERS

Machine Learning Institute

Feature Fiestas

- Helen Zhou
- Ewan Beattie
- Rosh Beed
- Aparna Pillai

Overfitting Overlords

- Joao Esteves
- Miguel Parracho
- Esperanza Shi
- Andrew

Hyperparameter Hippies

- Charles Cai
- Umut Sagir
- Peter O'Keeffe
- Ben Liong

Kernel Kittens

- Rasched Haidari
- Kadriye Turkcan
- Yali Pan
- David Edev

Recurrent Rebels

- Jingyan Chen
- Prima Gouse
- Anton Dergunov
- Marcin Tolysz

Perceptron Party

- Maria Sharif
- Dan Goss
- James Carter
- Felipe Lavratti

Dropout Disco

- Ben Bethell
- Tomas Krajcoviech
- Nikolas Kuhn
- Tao Zamorano

Backprop Bunch

- Jacob Jenner
- Andrei Zhirnov
- Adam Beedell
- Ethan Edwards

Gradient Gigglers

- Clement Ha
- Hikaru Tsujimura
- James Yan
- Melanie Wong

Bayesian Buccaneers

- Arjuna James
- Ben Williams
- Tyrone Nicholas