Reinforcement Learning with Human Feedback (ec101 (CSZZ4N)

Language Models can act as a century of rudimentary world models. They model agents, beliefs, and actions.

If my aim is to predict next token, you'd also want to predict agents' beliefs, background.

LMs as multitask assistants

zero-shot learning: no examples, no gradient updates.

GPT2 did better than SOTA supervised + finetuned in LAMBADA task.

GPT3:
in-context learning: examples of the task before your example.
(no gradient update)

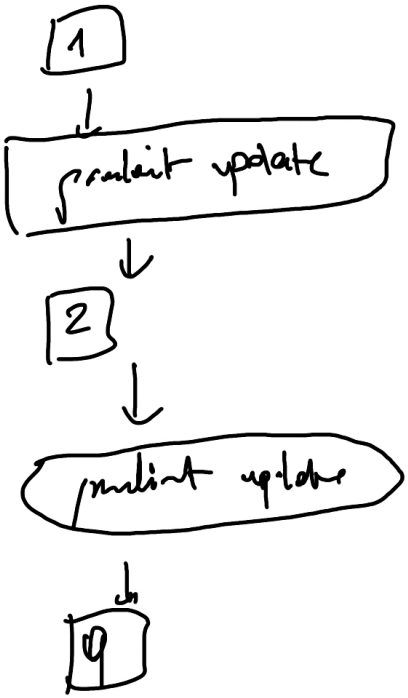
Zero-shot:

Translate English to French
cheese =

One-shot: ≈ 10% increase
Translate English to French
sea otter ⇒ loutre de mer
cheese =

Few-shot
couple of shots.

Traditional fine-tuning:



Few-shot learning:

Frozen model, no gradient updates.

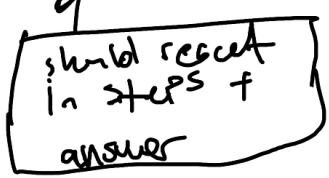
Chain of thought prompting:

standard

q
Answer
q
Answer

cot:

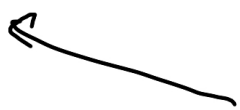
q
A with steps of reasoning.
q



Zero-shot cot prompting:

q:

Answer: Let's think step by step. + followed by step by step answering.



still manual cot is better though.

Language models aren't aligned with user intent.

Pretraining on language modeling.

↓

to train finetuning (on many tasks)
to allow the model to adapt to tasks.

Instruction finetuning:

Collect examples of (instruction, output) pairs across many tasks and finetune a Language Model.

evaluate on unseen tasks.

Multitask LMs. Massive Multitask Language Understanding Benchmark

↓

this isn't quite possible on creative generation tasks.

LM penalizes all token-level mistakes equally.
Some are more wrong than others.

for each sample from LM s_i let's assume we've $R(s) \in \mathbb{R}$

maximize Reward.

$E_{\hat{s} \sim P_\theta(s)}[R(\hat{s})]$ how do we change our LM params θ to maximize?

Policy gradient methods for maximizing parameters

Monte Carlo sampling:

The best way to approximate any expectation is to take samples and average them.

$$\theta_{t+1} : \theta_t + \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} R(s_i) \nabla \theta_t \cdot \log P_{\theta_t}(s_i)$$

if R is positively increasing, take gradient steps to maximize $P_\theta(s_i)$

if R is negative and decreasing, take steps to minimize $P_\theta(s_i)$

human-in-the-loop is expensive so modeling preferences is nice.

LM RM $\phi(s)$ to predict human preferences from a dataset and optimize to RM ϕ

Pairwise comparisons of deciding which option is more preferable also works. (Rate answers)

Reward model does come close to the single human preference if there's enough data.

$$\text{LM } p^{PT}(s) + RM_\phi(s) + \text{Policy Gradient}$$

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

Penalty when $p_\theta^{RL}(s) > p^{PT}(s)$

penalty: KL divergence between $p_\theta^{RL}(s)$ and $p^{PT}(s)$

Instruct GPT, scaling up RLHF to thousands of tasks. (30k tasks)