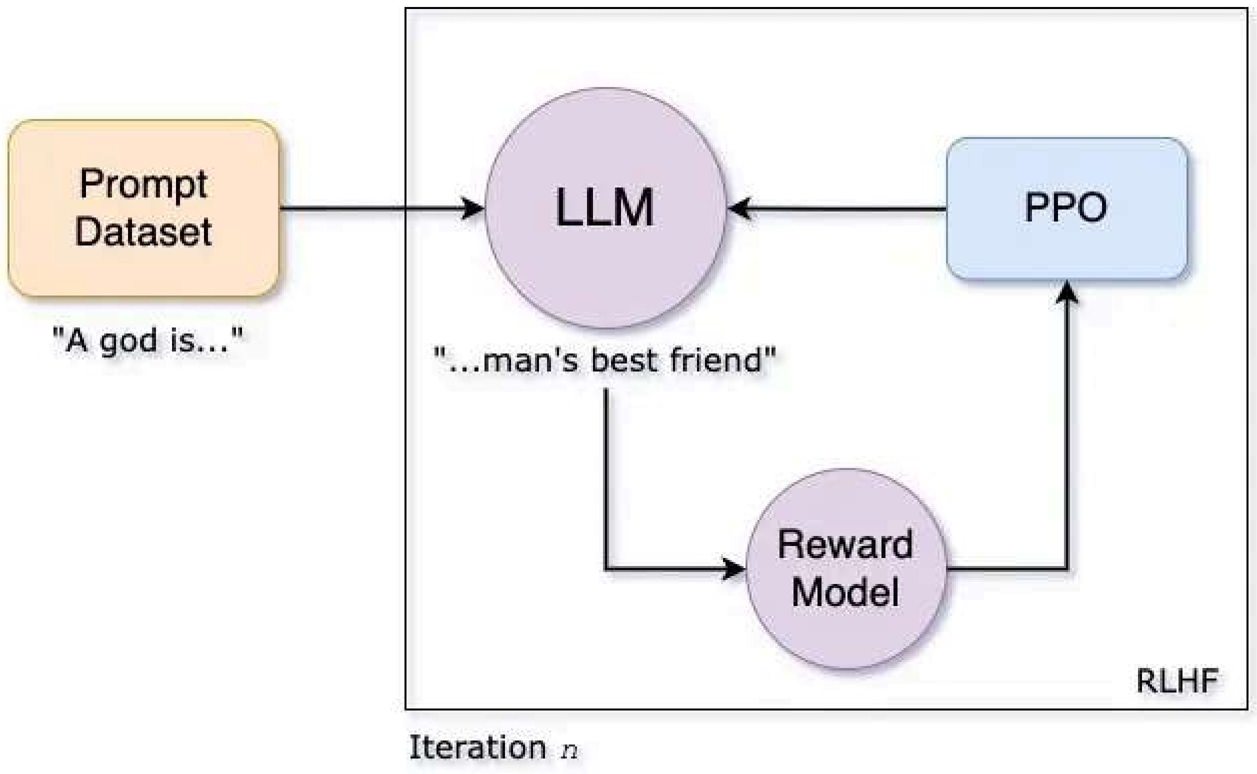
A Comprehensive Guide to Proximal Policy Optimization (PPO) in AI

Oleg Latypov · Follow

8 min read · Aug 28, 2023

19   2

> *OpenAI: We're releasing a new class of reinforcement learning algorithms, Proximal Policy Optimization (PPO), which perform comparably or better than state-of-the-art approaches while being much simpler to implement and tune. PPO has become the default reinforcement learning algorithm at OpenAI because of its ease of use and good performance.*



Simplifying Proximal Policy Optimization (PPO) for Language Models:

PPO, or Proximal Policy Optimization, is a smart technique used to solve problems related to teaching computers through `trial` and `error`. Think of it as a helpful method to train machines to understand and generate human-like text.

Here's how it works: Imagine you're training a computer program, like a virtual student, to write better and better essays. PPO helps this virtual student improve their essay-writing skills step by step.

Instead of making big changes all at once, PPO encourages small and gradual improvements. This way, the virtual student's writing doesn't change dramatically from one essay to the next. It's like refining their skills bit by bit without completely altering their style.

This cautious approach has a special name: **Proximal Policy Optimization**. “Proximal” means staying close to the original style, and “Policy Optimization” is about finding better strategies. By staying close to the original style, the virtual student’s improvements are more stable and consistent.

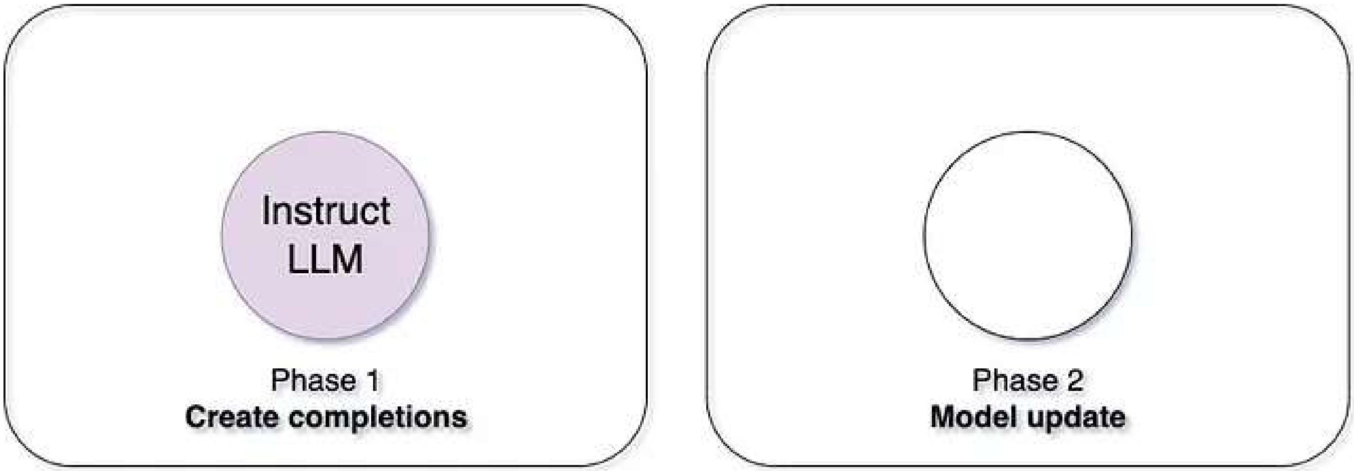
Now, let’s apply this to large language models (LLMs), which are advanced computer programs that understand and generate human-like language. PPO helps these models learn by making small adjustments to their language generation skills.

Imagine the LLM as a writer that’s learning to produce more engaging stories. Instead of suddenly changing the way it writes, PPO guides it to make slight improvements. This ensures that the new stories are similar to the previous ones, just a little better.

The ultimate goal is to teach the LLM to create stories that get the best response from readers. This is like helping the LLM become a skilled storyteller that knows how to capture people’s interest and imagination.

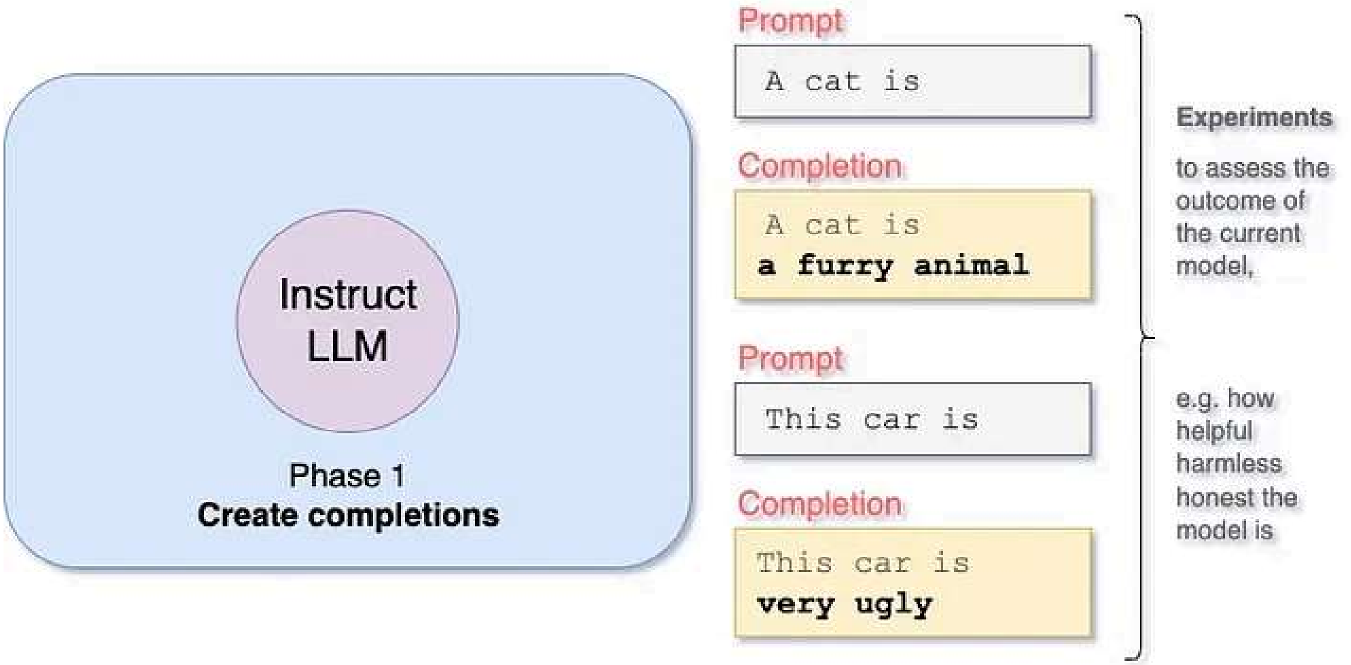
So, a PPO is like a patient teacher who helps computers improve their skills steadily and reliably. It’s a powerful technique, especially when training large language models to understand better and generate human-like language.

Let's discuss how this works in the specific context of large language models.



PPO Phase 1: **Create completions**

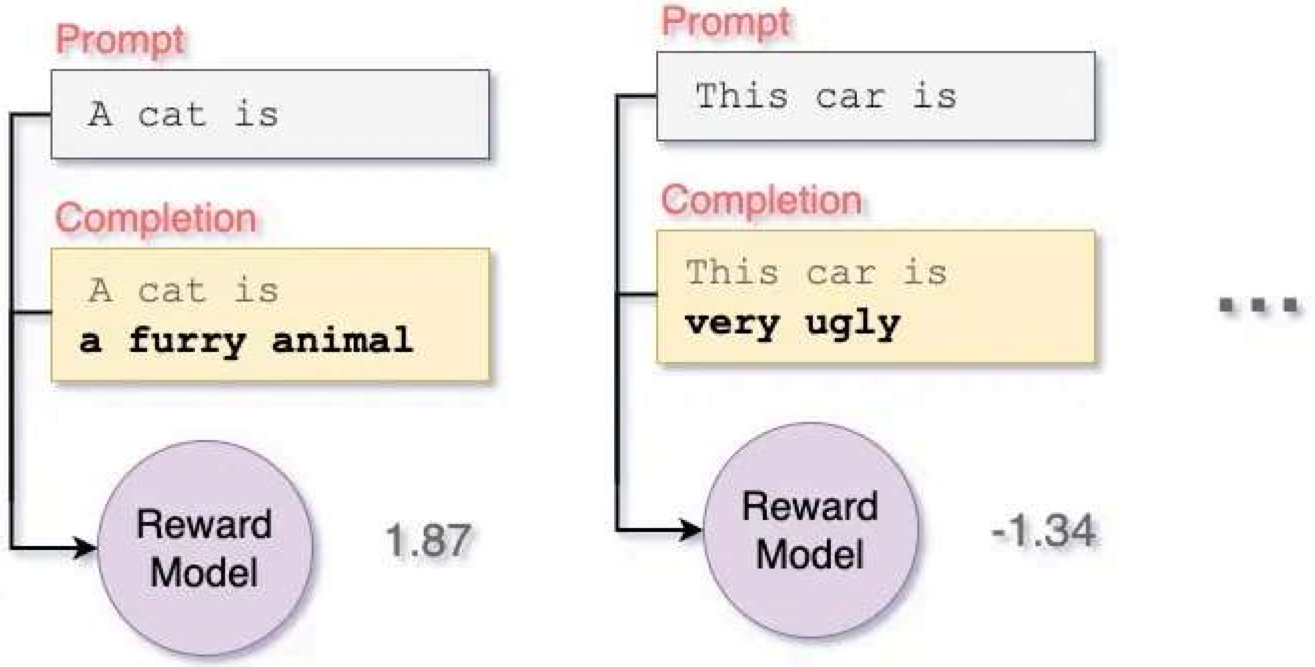
PPO begins with your initial instructions for the LLM. At a broad level, each PPO cycle involves two phases. In Phase I, the LLM performs several experiments by completing provided prompts. These experiments enable the updating of the LLM based on the reward model in Phase II.



PPO Phase 1: Create completions

Keep in mind that the reward model represents human preferences. For instance, the reward could determine the helpfulness, safety, and honesty of the responses. The anticipated reward of completion holds significance as a key factor in the PPO objective. This value is computed through a distinct component of the LLM known as the **value function** head.

Let's have a look at the value function and the value loss.



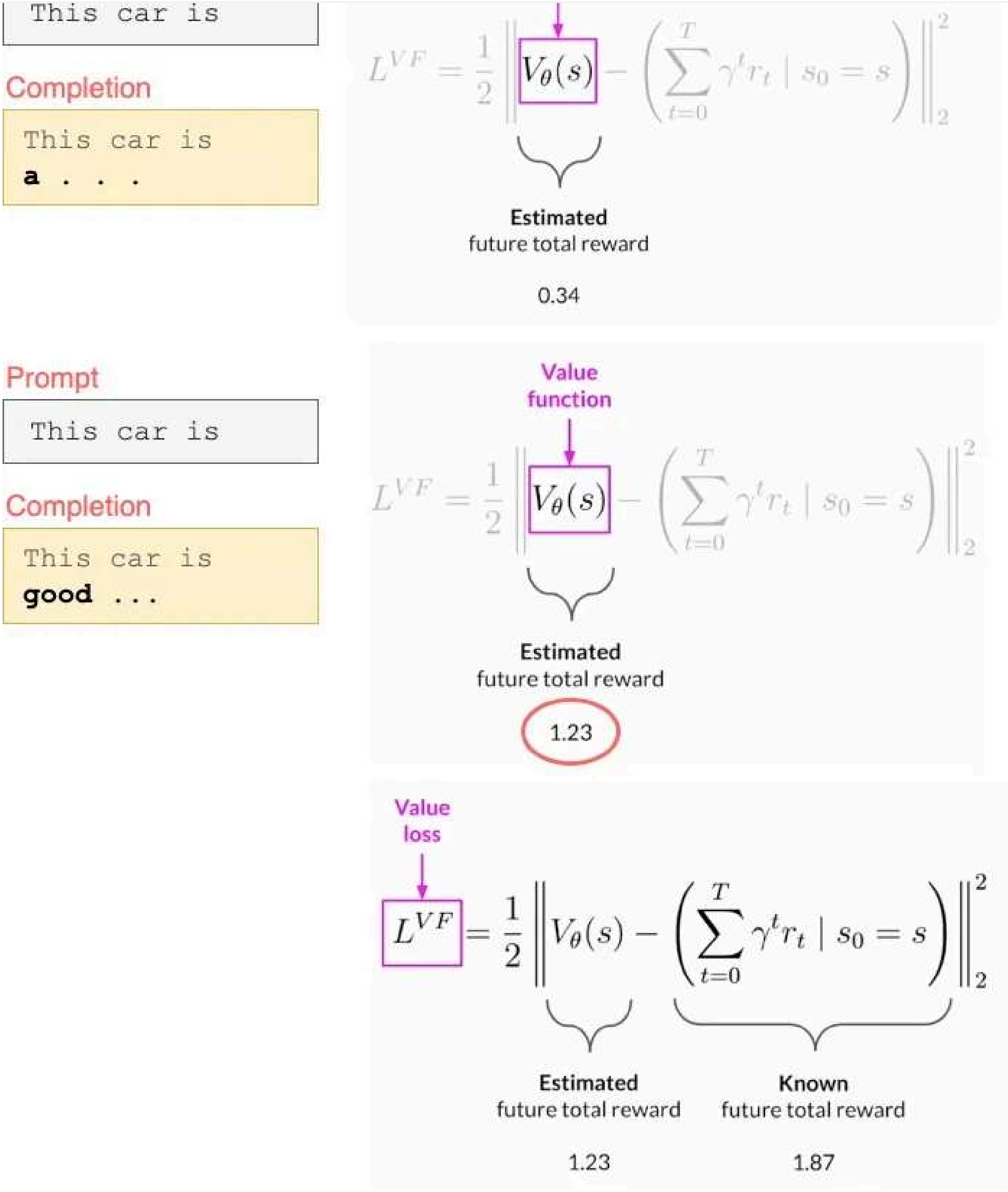
Rewards calculation

Imagine having a set of prompts. Initially, you generate responses using the LLM for these prompts. Following this, you assess the quality of these prompt completions using the reward model. For instance, one completion might earn a reward of 1.87, while another could get -1.34, and so forth. Each completion is paired with its corresponding reward.

Enter the value function: a tool within the LLM that predicts the overall expected reward for a specific State S. It works like this: as the LLM generates tokens for a completion, the value function estimates the total reward in the future, based on the existing sequence of tokens. This serves as a benchmark to judge the completion's quality against alignment standards.

For instance, let’s assume the estimated future reward at a certain point is 0.34. As the next token is generated, this estimate might rise to 1.23. The primary goal is to decrease value loss — the difference between the actual future reward (e.g., 1.87) and the value function’s approximation (e.g., 1.23) — thereby enhancing predictions for future rewards. This value function becomes integral in Phase 2’s Advantage Estimation process.

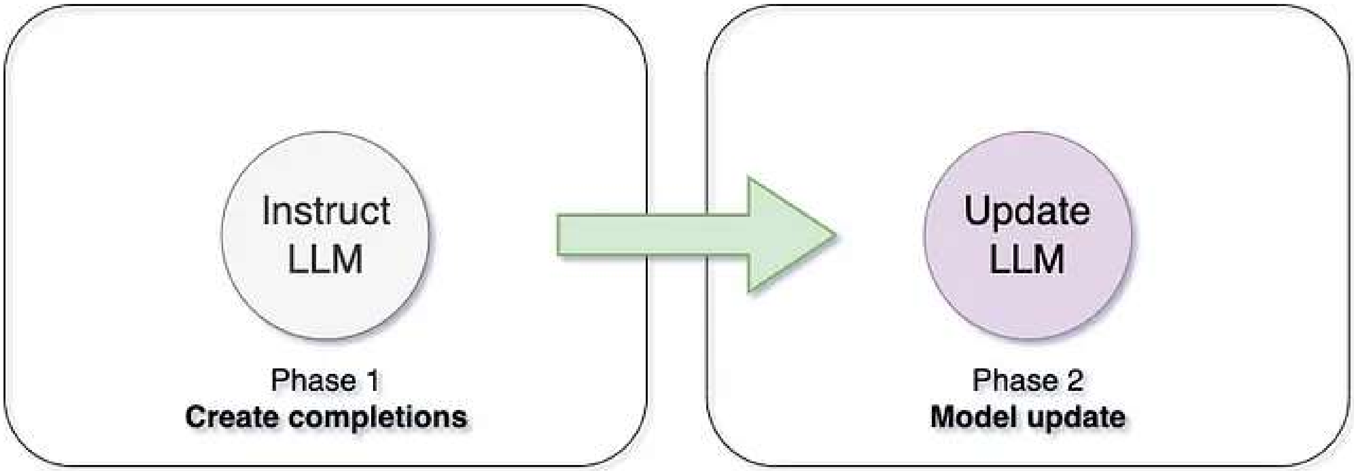
Calculate value loss



$$L^{VF} = \frac{1}{2}\left\| V_\theta(s) - \left(\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s\right) \right\|_2^2$$

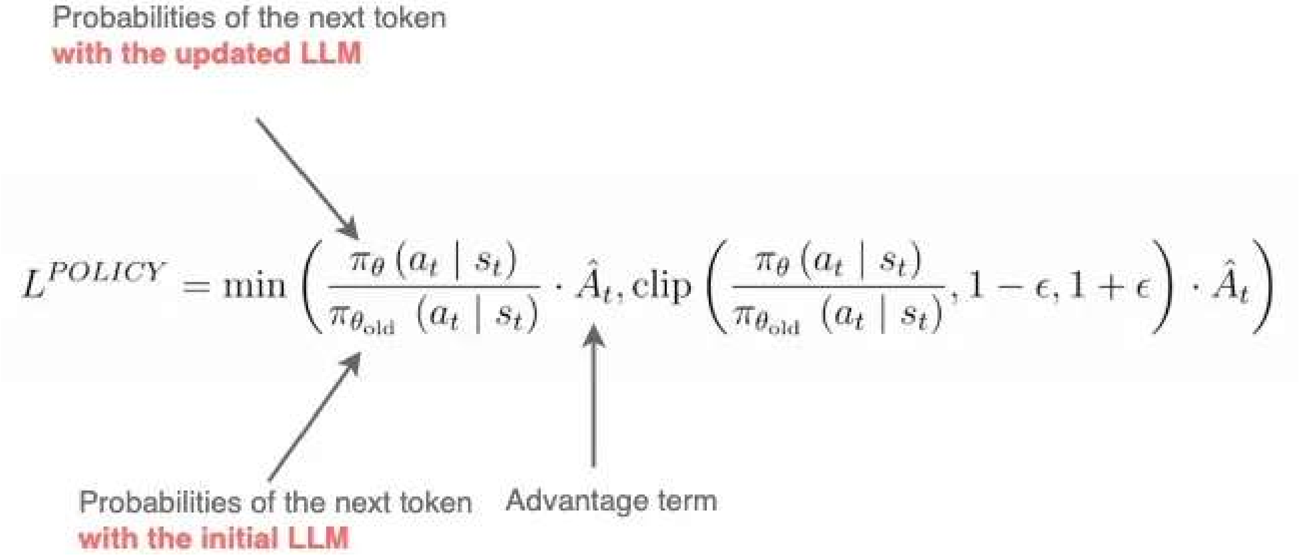
Calculate value loss

PPO Phase 2: Model update



PPO Phase 2: Model update

In Phase 2, we fine-tune the model by making small adjustments and measuring their impact on the model's alignment with its goals. These adjustments are based on how well the model responds to prompts and the resulting rewards and losses. PPO ensures that these adjustments stay within a defined range, which is an important aspect of the method. The goal is to gradually guide the model toward better outcomes. The core of PPO is its policy objective, which aims to create a strategy that leads to higher rewards. This means updating the model in a way that improves its responses to align more with what humans prefer, resulting in better rewards.

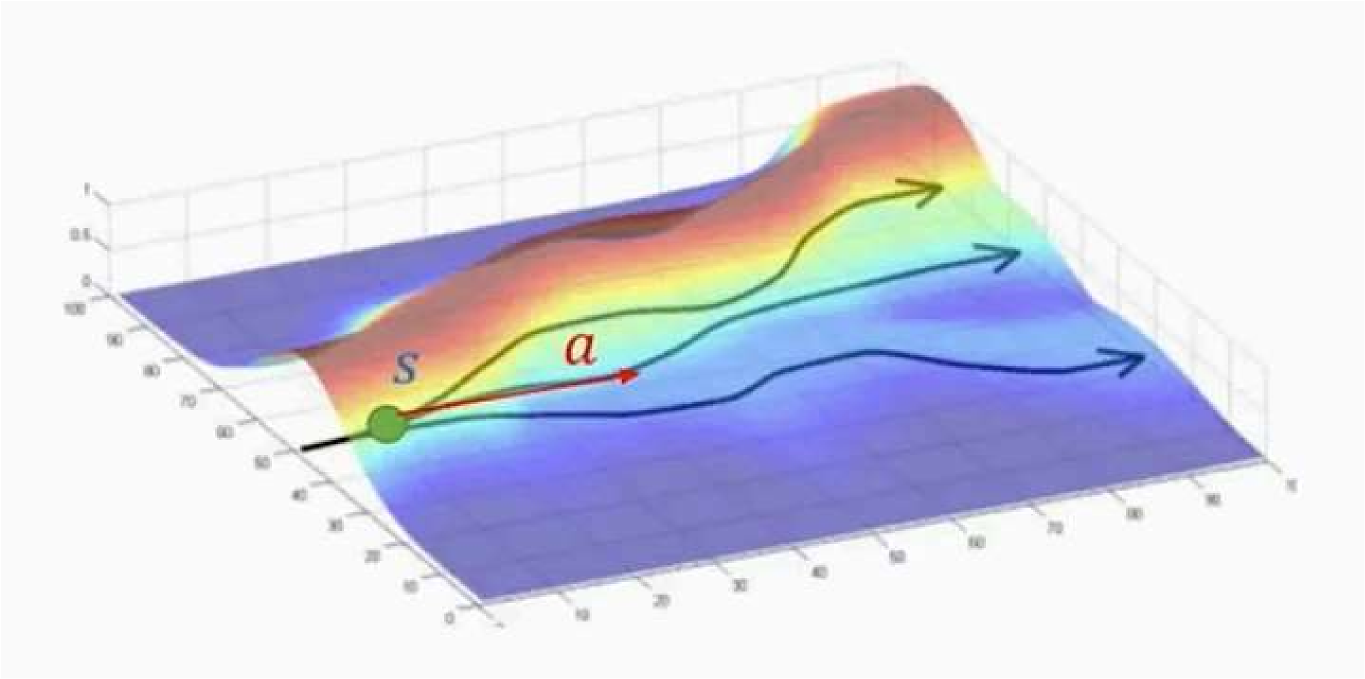


$$L^{POLICY} = \min\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \cdot \hat{A}_t, \text{clip}\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right)$$

Probabilities of the next token
with the updated LLM
Probabilities of the next token
with the initial LLM
Advantage term

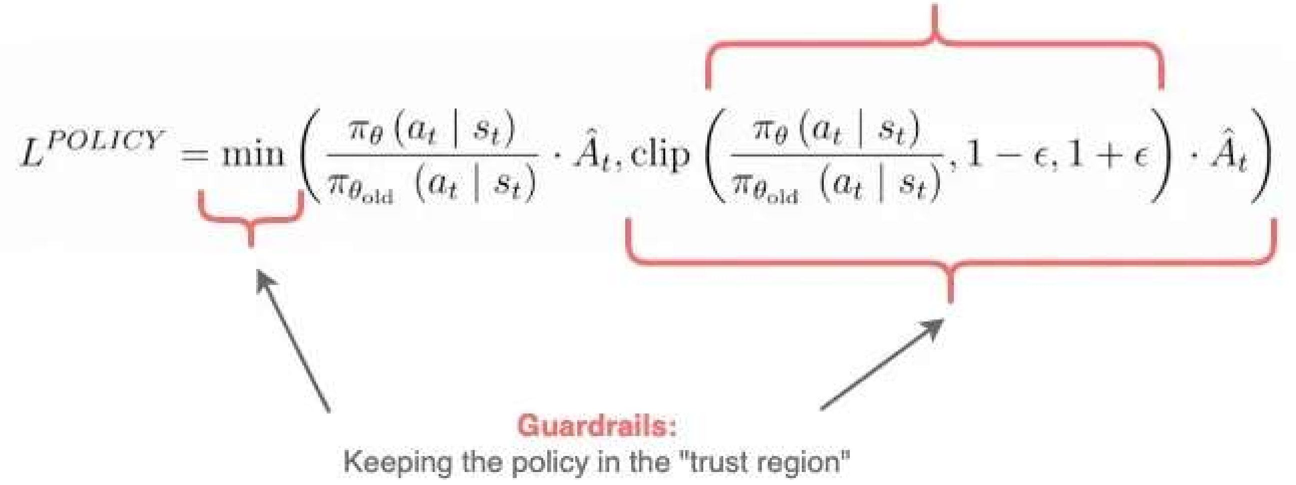
PPO Phase 2: Calculate policy loss

The policy loss takes the center stage in the PPO training process. Although the math might seem complex, it’s more straightforward than it appears. Let’s break it down step by step. Begin by paying attention to the essential part and setting aside the rest for now. In the context of an LLM, “Pi of A_t given S_t” refers to the probability of the next token (A_t) given the current prompt (S_t). The action represents the next token, and the state is the prompt completed up to that token (t). The denominator indicates the likelihood of the next token with the original, unchanged LLM. Meanwhile, the numerator reflects the probabilities of the next token using the updated LLM, which we can modify for better rewards. A-hat_t is referred to as the estimated advantage term for a particular action choice. This term gauges how much better or worse the current action is when compared to all potential actions at a given state of data. We delve into the anticipated future rewards of a completion that follows the new token, estimating how advantageous this completion stands in comparison to others. Although there’s a recursive formula for estimating this quantity based on the value function we discussed earlier, let’s focus on grasping the concept intuitively.



Let me illustrate what I've described visually. Imagine you have a prompt S, and there are various ways to complete it, depicted as different paths on the diagram. The advantage term serves as a guide, indicating whether the current token A_t is a better or worse choice compared to all the other potential tokens. In this visualization, consider the top path that rises higher — it signifies a better completion, resulting in a greater reward. On the contrary, the lower path that descends represents a worse completion.

> *So the overall conclusion is that maximizing this expression results in a better aligned LLM.*
>
> *Simply maximizing the expression can be problematic because our calculations rely on the assumption that our advantage estimations are accurate. These estimations are reliable when the old and new policies are similar. This is where the other terms in the equation become important.*



$$L^{POLICY} = \min\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \cdot \hat{A}_t, \operatorname{clip}\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right)$$

Guardrails:
Keeping the policy in the "trust region"

PPO Phase 2: Calculate policy loss

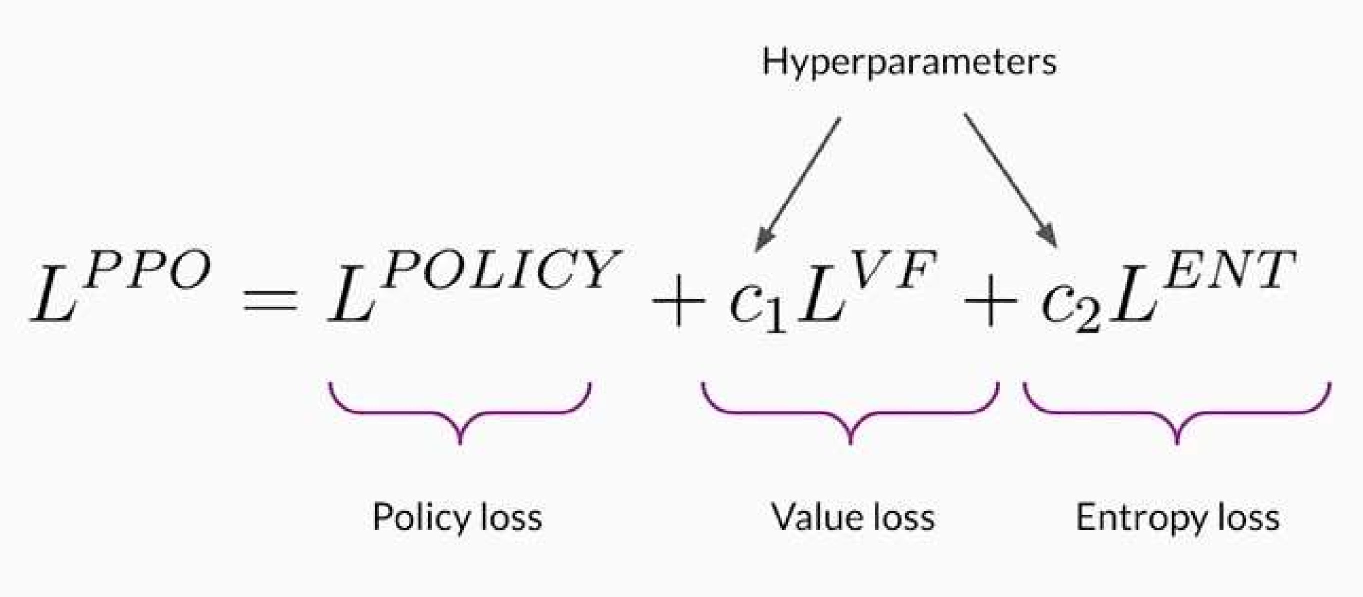
Looking at the equation again, we're essentially choosing the smaller value between two terms. One term we discussed earlier, and the other is a modified version that defines an area where two policies are close. This adjustment acts like guardrails, creating a safe zone around the LLM where our estimations remain accurate. This safe zone is called the “trust region,” and these adjustments prevent us from going too far from it.

In a nutshell, by optimizing the PPO policy objective, we enhance the LLM without risking inaccurate outcomes. This approach ensures that improvements stay within a reliable range.

Entropy loss

The policy loss steers the model toward its alignment goal, while entropy helps the model preserve creativity. If you minimize entropy, you could consistently generate prompts in the same manner as seen here. On the other hand, higher entropy encourages the LLM to be more imaginative.

The “temperature” affects the model’s creativity during inference, while “entropy” impacts creativity during training. When all these factors are combined with different weights, we arrive at our PPO objective. This objective updates the model to align more with human preferences in a consistent manner. It’s a comprehensive approach.



$$L^{PPO} = \underbrace{L^{POLICY}}_{\text{Policy loss}} + \underbrace{c_1 L^{VF}}_{\text{Value loss}} + \underbrace{c_2 L^{ENT}}_{\text{Entropy loss}}$$

PPO Phase 2: Objective function

The coefficients C1 and C2 are hyperparameters that fine-tune this process. The PPO objective adjusts model weights through backpropagation across multiple steps. After the weights are updated, a new PPO cycle begins. In each cycle, the model is replaced with the updated version, and this continues over many iterations. Gradually, this iterative process results in a model that is better aligned with human preferences.

Are there alternative reinforcement learning techniques applied to RLHF? Yes

Q-learning serves as another approach for refining LLMs through reinforcement learning, though currently, ***PPO stands as the favored method***. The prominence of PPO likely arises from its apt balance of complexity and performance. With that said, the process of enhancing LLMs via human or AI feedback remains a vigorously explored research domain, poised for further advancements in the immediate future. A case in point is a recent publication by Stanford researchers introducing "direct preference optimization," a simpler alternative to RLHF. While these nascent methods are actively maturing, more investigation is warranted to comprehensively ascertain their advantages. In my view, this research realm holds a tremendous sense of excitement and potential.

In essence, this topic may present itself as intricate and multifaceted; however, it holds immense importance for those seeking an in-depth comprehension of AI.

Thank you for reaching this juncture.

AI OpenAI ChatGPT Data Science