4 Training Overview

4.1 Problem Formulation

The optimization of reinforcement learning from human feedback (RLHF) builds on top of the standard RL setup. In RL, an agent takes actions, a, sampled from a policy, π, with respect to the state of the environment, s, to maximize reward, r [54]. Traditionally, the environment evolves with respect to a transition or dynamics function $p(s_{t+1}|s_t, a_t)$. Hence, across a finite episode, the goal of an RL agent is to solve the following optimization:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right], \tag{6}$$

where γ is a discount factor from 0 to 1 that balances the desirability of near- versus future-rewards. Multiple methods for optimizing this expression are discussed in Chapter 11.

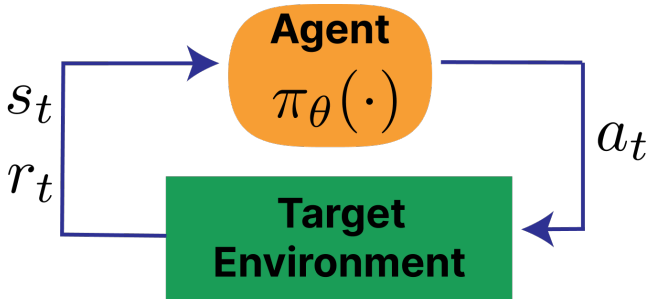

Figure 2: Standard RL loop

A standard illustration of the RL loop is shown in fig. 2 and how it compares to fig. 3.

4.2 Manipulating the Standard RL Setup

There are multiple core changes from the standard RL setup to that of RLHF:

1. Switching from a reward function to a reward model. In RLHF, a learned model of human preferences, $r_\theta(s_t, a_t)$ (or any other classification model) is used instead of an environmental reward function. This gives the designer a substantial increase in the flexibility of the approach and control over the final results.
2. No state transitions exist. In RLHF, the initial states for the domain are prompts sampled from a training dataset and the "action" is the completion to said prompt. During standard practices, this action does not impact the next state and is only scored by the reward model.
3. Response level rewards. Often referred to as a bandit problem, RLHF attribution of reward is done for an entire sequence of actions, composed of multiple generated tokens, rather than in a fine-grained manner.

Given the single-turn nature of the problem, the optimization can be re-written without the time horizon and discount factor (and the reward models):

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[r_\theta(s_t, a_t)\right]. \tag{7}$$

In many ways, the result is that while RLHF is heavily inspired by RL optimizers and problem formulations, the action implementation is very distinct from traditional RL.

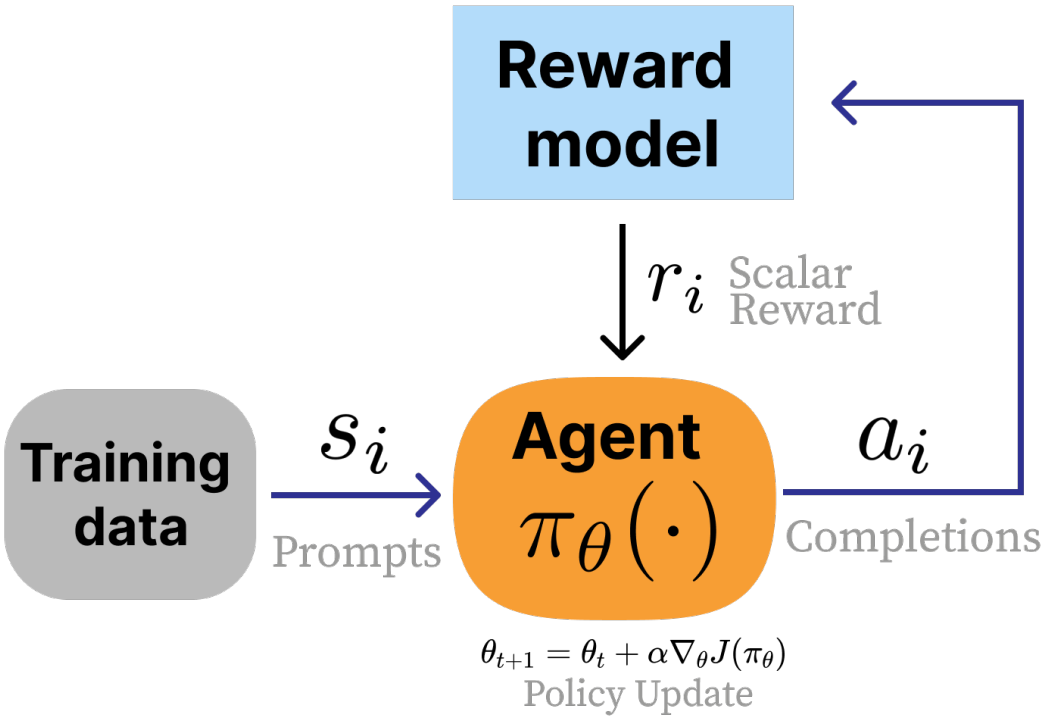

Figure 3: Standard RLHF loop

4.3 Optimization Tools

In this book, we detail many popular techniques for solving this optimization problem. The popular tools of post-training include:

- **Reward modeling** (Chapter 7): Where a model is trained to capture the signal from collected preference data and can then output a scalar reward indicating the quality of future text.
- **Instruction finetuning** (Chapter 9): A prerequisite to RLHF where models are taught the question-answer format used in the majority of language modeling interactions today by imitating preselected examples.
- **Rejection sampling** (Chapter 10): The most basic RLHF technique where candidate completions for instruction finetuning are filtered by a reward model imitating human preferences.
- **Policy gradients** (Chapter 11): The reinforcement learning algorithms used in the seminal examples of RLHF to update parameters of a language model with respect to the signal from a reward model.
- **Direct alignment algorithms** (Chapter 12): Algorithms that directly optimize a policy from pairwise preference data, rather than learning an intermediate reward

model to then optimize later.

Modern RLHF-trained models always utilize instruction finetuning followed by a mixture of the other optimization options.

4.4 RLHF Recipe Example

The canonical RLHF recipe circa the release of ChatGPT followed a standard three step post-training recipe where RLHF was the center piece [55] [3] [5]. The three steps taken on top of a "base" language model (the next-token prediction model trained on large-scale web text) was, summarized below in fig. 4:

1. **Instruction tuning on ~10K examples**: This teaches the model to follow the question-answer format and teaches some basic skills from primarily human-written data.
2. **Training a reward model on ~100K pairwise prompts**: This model is trained from the instruction-tuned checkpoint and captures the diverse values one wishes to model in their final training. The reward model is the optimization target for RLHF.
3. **Training the instruction-tuned model with RLHF on another ~100K prompts**: The model is optimized against the reward model with a set of prompts that the model generates over before receiving ratings.

Once RLHF was done, the model was ready to be deployed to users. This recipe is the foundation of modern RLHF, but recipes have evolved substantially to include more stages and more data.

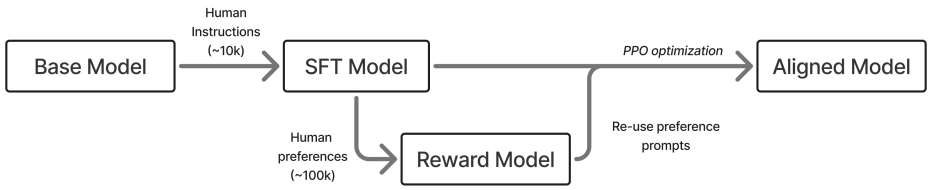

Figure 4: A rendition of the early, three stage RLHF process with SFT, a reward model, and then optimization.

Modern versions of post-training involve many, many more model versions. An example is shown below in fig. 5 where the model undergoes numerous training iterations before convergence.

4.5 Finetuning and Regularization

RLHF is implemented from a strong base model, which induces a need to control the optimization from straying too far from the initial policy. In order to succeed in a finetuning regime, RLHF techniques employ multiple types of regularization to control the optimization. The most common change to the optimization function is to add a distance penalty on

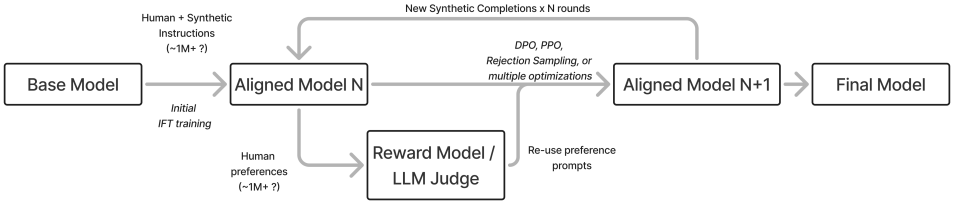

Figure 5: A rendition of modern post-training with many rounds.

the difference between the current RLHF policy and the starting point of the optimization:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[r_\theta(s_t, a_t)\right] - \beta\mathcal{D}_{KL}(\pi^{\text{RL}}(\cdot|s_t)\|\pi^{\text{ref}}(\cdot|s_t)). \tag{8}$$

Within this formulation, a lot of study into RLHF training goes into understanding how to spend a certain "KL budget" as measured by a distance from the initial model. For more details, see Chapter 8 on Regularization.