

## Iterative Fine-Tuning

As we received more batches of human preference data annotation, we were able to train better reward models and collect more prompts. We therefore trained successive versions for RLHF models, referred to here as RLHF-V1, RLHF-V5.

We explored RLHF fine-tuning with two main algorithms:

- **Proximal Policy Optimization (PPO)** (Schulman et al., 2017), the standard in RLHF literature.
- **Rejection Sampling fine-tuning.** We sample  $K$  outputs from the model and select the best candidate with our reward, consistent with Bai et al. (2022b). The same re-ranking strategy for LLMs was also proposed in Deng et al. (2019), where the reward is seen as an energy function. Here, we go one step further, and use the selected outputs for a gradient update. For each prompt, the sample obtaining

**Figure 7: Max and median reward among  $N$  samples**,  $N \in [1, \dots, 100]$  averaged over our training set of prompts. The delta between max and median can be interpreted as potential gain with Rejection Sampling.

the highest reward score is considered the new gold standard. Similar to Scialom et al. (2020a), we then fine-tune our model on the new set of ranked samples, reinforcing the reward.

The two RL algorithms mainly differ in:

- *Breadth* — in Rejection Sampling, the model explores  $K$  samples for a given prompt, while only one generation is done for PPO.
- *Depth* — in PPO, during training at step  $t$  the sample is a function of the updated model policy from  $t - 1$  after the gradient update of the previous step. In Rejection Sampling fine-tuning, we sample all the outputs given the initial policy of our model to collect a new dataset, before applying the fine-tuning similar to SFT. However, since we applied iterative model updates, the fundamental differences between the two RL algorithms are less pronounced.

Until RLHF (V4), we used only Rejection Sampling fine-tuning, and after that, we combined the two sequentially, applying PPO on top of the resulted Rejection Sampling checkpoint before sampling again.

**Figure 8: RLHF impact of the temperature** when sampling  $N$  outputs and scoring them with a reward model.

**Rejection Sampling.** We perform rejection sampling only with our largest 70B Llama 2-Chat. All smaller models are fine-tuned on rejection sampled data from the larger model, thus distilling the large-model capabilities into the smaller ones. We leave further analysis of the effect of this distillation for future work.

At each iterative stage, we sample  $K$  answers for each prompt from the most recent model. We score each sample given the best reward model accessible at the time of the experiment, and then select the best answer for a given prompt. In earlier versions of our model, up to RLHF V3, our approach was to confine answer selection solely to the “bag” of samples gathered from the preceding iteration. For example, RLHF V3 was trained using only samples from RLHF V2. However, despite continuous improvement, this method led to a regression in some capabilities. For example, RLHF V3 struggled more than previous versions to compose rhyming lines in poems, as discerned through qualitative analysis, suggesting that further investigation into the causes of and mitigations for forgetting (Kirkpatrick et al., 2017; Nguyen et al., 2019; Ramasesh et al., 2021) could be a fruitful area for additional future research.

In response, on subsequent iterations, we modified our strategy, incorporating top-performing samples from all prior iterations, such as those used in RLHF-V1 and RLHF-V2. Although we do not present specific figures, this adjustment demonstrated considerable enhancements in performance and effectively addressed the previously noted issues. This mitigation can be seen as analogous to Synnaeve et al. (2019) and Vinyals et al. (2019) in the RL literature.

We illustrate the benefit of Rejection Sampling in Figure 7. The delta between the maximum and median curves can be interpreted as the potential gain of fine-tuning on the best output. As expected, this delta increases with more samples, since the maximum increases (i.e., more samples, more opportunities to generate a good trajectory), while the median remains stationary. There is a direct connection between the exploration and the maximum reward we can obtain among the samples. The temperature parameter also plays an important role for exploration, as a higher temperature enables us to sample more diverse outputs.

In Figure 8, we report for a Llama 2-Chat-SFT (left) and a Llama 2-Chat-RLHF (right), the maximum reward curves among  $N$  samples (with  $N \in [1, \dots, 100]$ ), for different temperatures. We can observe that the optimal temperature is not constant during the iterative model updates: RLHF has a direct impact on rescaling the temperature. For Llama 2-Chat-RLHF, the optimal temperature when sampling between 10 and 100 outputs is  $T \in [1.2, 1.3]$ . Given a finite compute budget, it is therefore necessary to re-adjust the temperature progressively. Note that this temperature rescaling happens for a constant number of steps for each model, and always starting from the base model on each new RLHF version.

**PPO.** We further train our language model following the RL scheme of Stiennon et al. (2020), which uses the reward model as an estimate for the true reward function (human preference) and the pretrained language model as the policy to optimize.

We iteratively improve the policy by sampling prompts  $p$  from our dataset  $D$  and generations  $g$  from the policy  $\pi$  and use the PPO algorithm and loss function to achieve this objective.

contains a penalty term for diverging from the original policy  $\pi_0$ . As was observed in other works (Stiennon et al., 2020; Ouyang et al., 2022), we find this constraint is useful for training stability, and to reduce reward hacking whereby we would achieve high scores from the reward model but low scores from human evaluation.

We define  $R_c$  to be a piecewise combination of the safety ( $R_s$ ) and helpfulness ( $R_h$ ) reward models. We have tagged prompts in our dataset that might elicit potentially unsafe responses and prioritize the scores from the safety model. The threshold of 0.15 is chosen for filtering unsafe responses, corresponding to a precision of 0.89 and a recall of 0.55 evaluated on the Meta Safety test set. We also find it important to whiten the final linear scores (shown here by reversing the sigmoid with the logit function) in order to increase stability and balance properly with the KL penalty term ( $\beta$ ) above.

We train for between 200 and 400 iterations for all our models, and use evaluations on held-out prompts for early stopping. Each iteration of PPO on the 70B model takes on average  $\approx 330$  seconds. To train quickly with large batch sizes, we use FSDP (Zhao et al., 2023). This was effective when using  $O(1)$  forward or backward passes, but caused a large slow down ( $\approx 20\times$ ) during generation, even when using a large batch size and KV cache. We were able to mitigate this by consolidating the model weights to each node once before generation and then freeing the memory after generation, resuming the rest of the training loop.