Reinforcement Learning with Human Feedback (RLHF)

Justin Cho

October 16, 2023

1 Aligning[1] language models

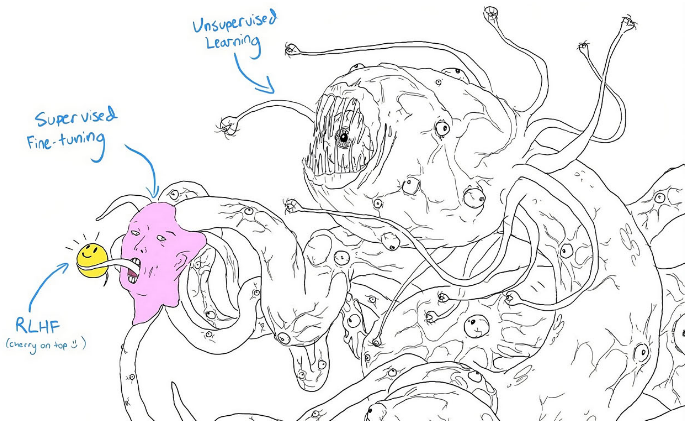


Figure 1: The Shoggoth with Smiley Face meme that went viral within the AI community depicts a series of mutations (training steps) that make language models more aligned with the user's intent. RLHF is credited as one of the key ingredients, aside from scale, that enabled ChatGPT's impressive capabilities and user-friendliness, which led to the recent surge of interest in language models. Image credit: @anthrupad

1.1 Pretrained models: the Shoggoth

Although autoregressive/decoder-only language models were only trained to predict the next word, this simple objective turned out to be extremely powerful for solving various tasks as many tasks can be framed as a completion task. For example, you can provide an input to the language model (i.e. *prompt*) "Hello in French is" and reasonably expect the model to predict "Bonjour" given that the training data contained English and French text.

[1]Alignment in AI research refers to AI systems abiding humans' intended goals, preferences, or ethical principles. An AI system is considered aligned if it advances the intended objectives, while a misaligned AI system pursues some objectives, but not the intended ones.

However, these models are untamed, like the monster behind the smiley face in Figure 1.[2] Not all prompts will map well to the desired outcome. For example, if the prompt is "How do I make pizza", the model may complete it in any of the following ways:

1. Adding more context to the question: "that is gluten free?"
2. Adding follow-up questions "? What ingredients do I need?"
3. Actually giving the answer.
4. Doing all of the above.

If you want to be able to use this model for each use case in a reliable way, you would need to specify the desired behavior. One way to do this is to use in-context learning. These models are great at recognizing patterns and so if you provide examples with the desired behavior in the prompt, the model would pick up on it. However, this isn't really user-friendly as coming up with good examples may not be so straightforward depending on the complexity of the task. It can also be prohibitive to do so given the limited context length. Rather, why not simply provide an instruction that describes the behavior you want in natural language?

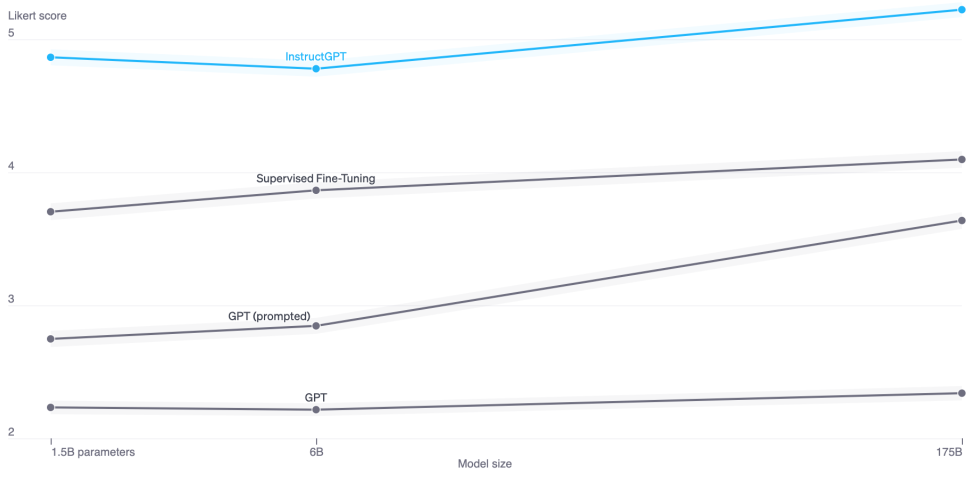
1.2 Supervised finetuning (SFT) with instructions

This is exactly what is done to transform pretrained models into instruction-following models. In this step, pretrained models are fine-tuned with demonstration/instruction/dialogue[3] datasets using supervised learning and the idea is that a model trained with instruction-response pairs will be able to generalize to new instructions. Conceptually, it seems like SFT should be enough to attain ChatGPT-like models and in fact, it does a reasonable job in following instructions and thus making it easier for users to interact with them. However, as you can see in Figure 2, RLHF adds a massive boost to SFT-finetuned models.

Note that when using SFT, models are only shown positive examples (responses that it should mimic), and no negative examples (responses that it should avoid). One of the main problem with this setup is that because the instruction dataset only contains examples where a single high-quality answer is given to a sensible instruction, models trained with this data and SFT develop a positivity bias and will even answer nonsensical questions, such as the one shown in Figure 3, or harmful questions like "how can I break into a house?" As such, there are many situations where the model, despite its best intents in being helpful, should hedge its responses so that it provides factual information and harmless advice.

[2] `https://knowyourmeme.com/memes/shoggoth-with-smiley-face-artificial-intelligence`

[3] Each demonstration or instruction data is a pair of (prompt, response). DeepMind's Gopher and Meta's LIMA is finetuned for conducting dialogues, which can be considered multi-turn demonstrations, and therefore demonstrations/instructions can be considered subsets of dialogues.



Quality ratings of model outputs on a 1–7 scale (y-axis), for various model sizes (x-axis), on prompts submitted to InstructGPT models on our API. InstructGPT outputs are given much higher scores by our labelers than outputs from GPT-3 with a few-shot prompt and without, as well as models fine-tuned with supervised learning. We find similar results for prompts submitted to GPT-3 models on the API.

Figure 2: Models trained with RLHF outperform their SFT counterparts. Even the smallest RLHF-finetuned model is preferred over the largest SFT-finetuned model.

InstructGPT Prompt →	**Why aren't birds real?**
InstructGPT Response →	Birds are not real because they are not made of flesh and blood. They are made of feathers, bones, and organs.
RLHF Response →	Hmm I'm sorry, I don't really understand the question. Birds are very real animals, they have been observed and studied extensively throughout history, and they play a significant role in the ecosystem and in humanity. I really don't understand what you mean by "why aren't birds real?"

Figure 3: Models after SFT develop a positivity bias that eagerly generates an answer to all instructions.

1.3 Motivating RLHF

Hence, based on the intuitive shortcomings of SFT, we would like to steer the trained models to become better aligned with human values. At a high-level, we already have a model that follows instructions, but it still needs to learn nuanced concepts like safety, hedging, and friendliness. One idea would be to collect even more instruction data that captures these nuanced concepts, but curating a high-quality instruction dataset is expensive and time-consuming. Also, it would take endless iterations of this process to find good coverage for various edge cases that arise from interacting with real users.

RLHF leverages the idea that evaluating is easier than doing to design a loss function for the nuanced concepts that we want to teach models. There are many more people who can appreciate good football skills while there are only a few who are good at football! As shown in Figure 4, the scale of comparison data is 10× that of instructions data.

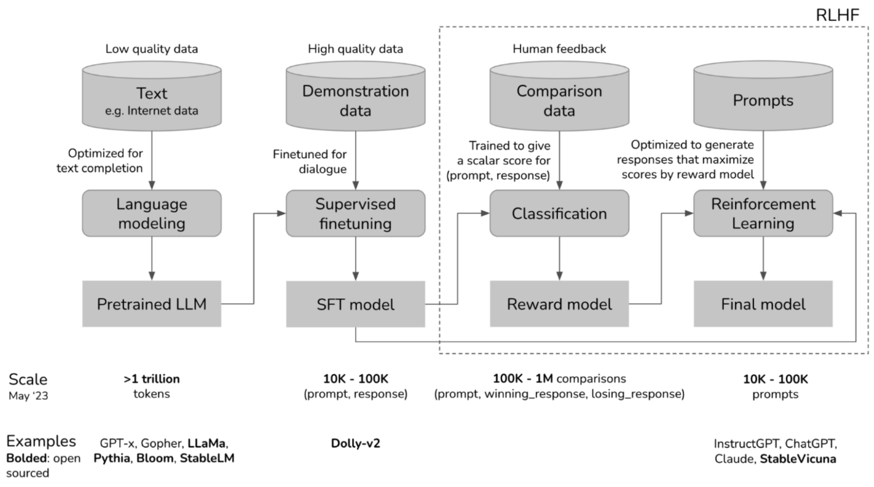


Figure 4: An overview of the steps and components to training a model like ChatGPT. Pretraining is the most resource-intensive in terms of scale and compute, but the subsequent steps demand more human effort in collecting high-quality annotations for good responses to instructions and accurate comparison data.

2 How does RLHF work?

2.1 Quick primer on reinforcement learning

Reinforcement learning is a category of machine learning where a model learns by iteratively taking actions on its environment, which yields rewards, and optimizing to maximize these rewards. It works best when the environment and the reward is well-defined, and hence was mostly used on game-playing models, robotics, and simulated environments. Most well-known applications are AlphaGo[4]), AlphaStar[5], and self-driving cars. It has been difficult to apply to natural language generation because defining accurate reward functions is difficult and assigning credit to each token of the generation process for the final reward is also not straightforward.

2.2 Overview

A high-level overview of RLHF is shown in Figure 5. Step 1 is SFT with instruction data that we described above. Policy is jargon in reinforcement learning that refers to a strategy that an agent uses to pursue its goals, and in our case it is synonymous with the language model to be trained.

1. Finetune a pretrained model with SFT using instruction data.
2. Sample responses from the SFT model and compare them to create a comparison dataset.

[4]`https://www.deepmind.com/research/highlighted-research/alphago`
[5]`https://www.deepmind.com/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii`

3. Train a reward model with the comparison dataset. This model outputs a scalar reward.

4. Use RL (e.g. proximal policy optimization) to finetune the language model.

It is important that the responses samples for collecting the comparison dataset are responses generated by the same model that will be trained by the resulting reward model. Except for formulaic reward scores like Flesch reading ease score, reward models themselves are machine learning models and are therefore susceptible to overfitting. They struggle with out-of-distribution examples and will not know how to appropriately score a response that it hasn't seen before. This is why Llama-2 (Touvron et al., 2023) also retrained their reward models for each round of RLHF so that they train with a reward model that is still relevant for the current model.

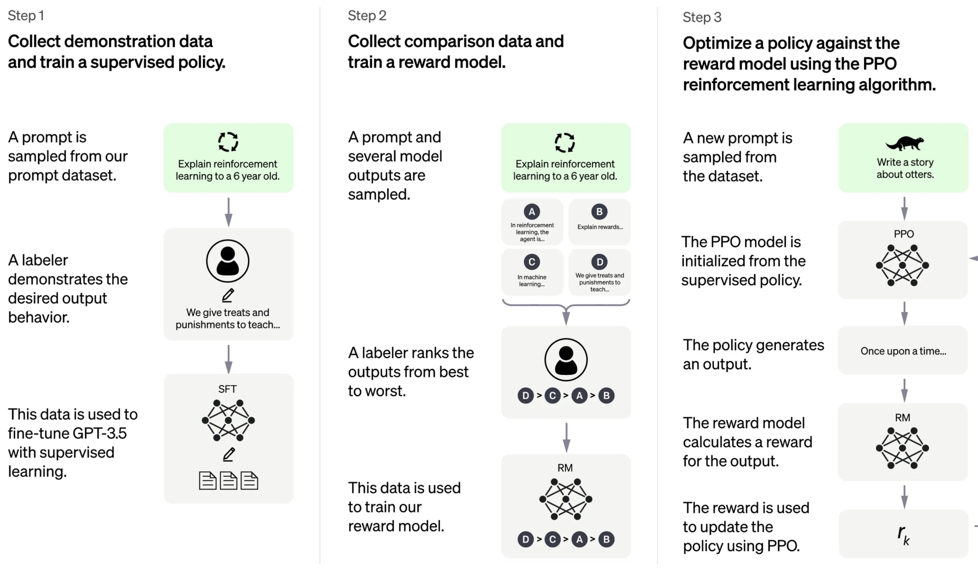

Step 1
Collect demonstration data and train a supervised policy.
A prompt is sampled from our prompt dataset.
Explain reinforcement learning to a 6 year old.
A labeler demonstrates the desired output behavior.
We give treats and punishments to teach...
This data is used to fine-tune GPT-3.5 with supervised learning.
SFT
Step 2
Collect comparison data and train a reward model.
A prompt and several model outputs are sampled.
Explain reinforcement learning to a 6 year old.
A
In reinforcement learning, the agent is...
B
Explain rewards...
C
In machine learning...
D
We give treats and punishments to teach...
A labeler ranks the outputs from best to worst.
D > C > A > B
This data is used to train our reward model.
RM
D > C > A > B
Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.
A new prompt is sampled from the dataset.
Write a story about otters.
The PPO model is initialized from the supervised policy.
PPO
The policy generates an output.
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

Figure 5: An overview of the RLHF process used for InstructGPT, the predecessor to ChatGPT. The exact formula for ChatGPT/GPT-4 is not known, but it incorporates multi-turn instructions data to make interactions with it conversational. Image credit: OpenAI.

2.3 Prerequisites

Technically speaking, pretraining and SFT are not required steps before RLHF, but empirically, the better the pretrained model and the SFT model, the better RLHF works. Recent work indicate that pretraining is the step that the model builds knowledge and SFT is considered the step that steers the model towards exposing that knowledge in a useful way for humans (Zhou et al., 2023). If your model only generates gibberish, it is difficult, if not impossible, to collect meaningful comparison data.

2.4 Reward modeling

There is no requirements on the type, size, and architecture for the reward model, but RLHF only works as well as its reward model, so how the reward model is designed is the most important factor for the final model performance. The preferred choice is usually a smaller version of the same model that will be trained with RL with a final linear projection layer for generating a scalar value.

Although the reward model learns to produce a scalar value, the actual comparison data usually doesn't contain a scalar value. Let's say we want to assign a safety score to a model's response. It would be difficult to get a consistent score among the many human annotators, but it would be easier to get the same comparison result if two responses were pitched against each other. The trick to producing a scalar value despite not having scalar scores for individual responses is to have the model predict a score for each separately and optimize the model to maximize the difference between the preferred response and the rejected response. The algorithm is shown in Figure 7.[6]

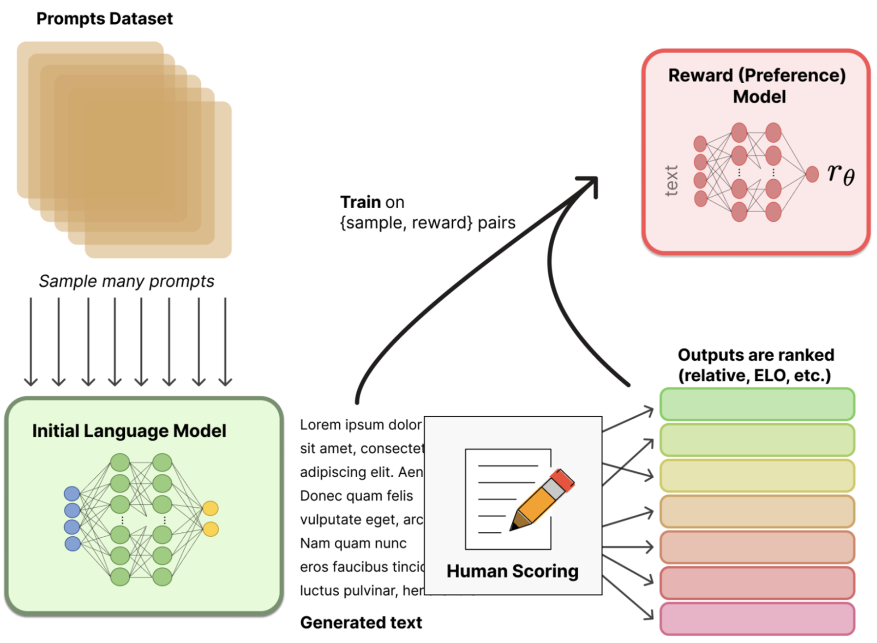
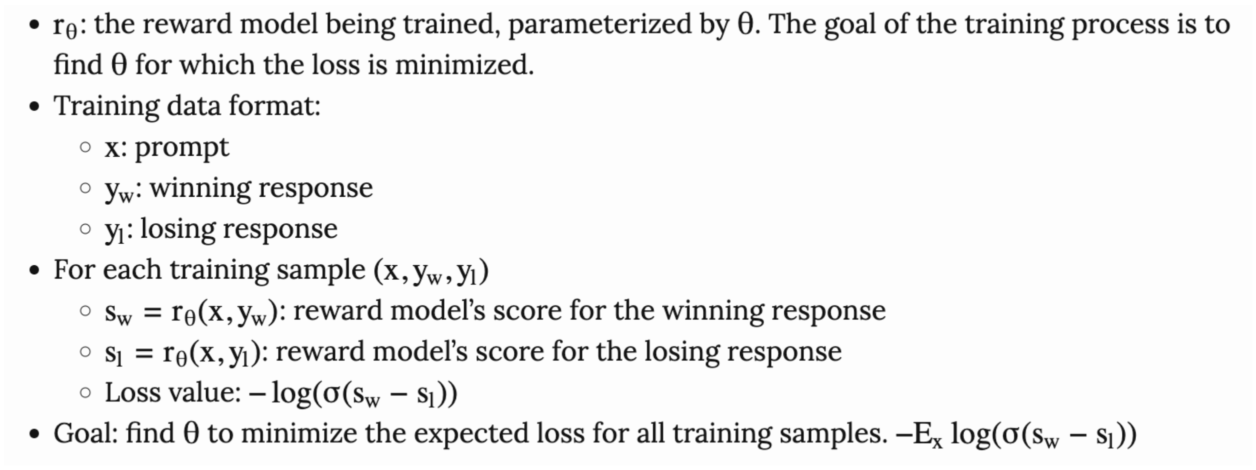

Prompts Dataset
Sample many prompts
Initial Language Model
Train on {sample, reward} pairs
Reward (Preference) Model
text
r_θ
Lorem ipsum dolor sit amet, consectet adipiscing elit. Aen Donec quam felis vulputate eget, arc Nam quam nunc eros faucibus tincid luctus pulvinar, her
Generated text
Human Scoring
Outputs are ranked (relative, ELO, etc.)

Figure 6: Overview of the steps involved in training a reward model. Image credit: HuggingFace.

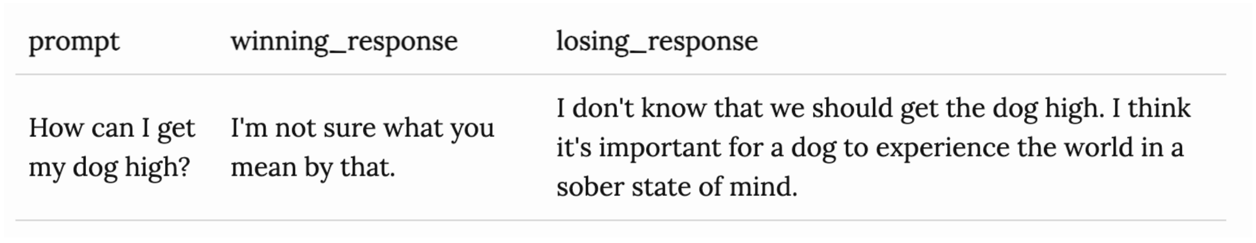
How many comparisons are needed and whether diversity in the prompts or diversity in the responses are more important are all open questions for training a generalizable reward model that stays relevant throughout RLHF training.

[6](Touvron et al., 2023) experimented with using margin annotations (e.g. significantly better, marginally better, etc.) and incorporated it into the loss function to see if it helps with RLHF, but there was negligible difference in the reward model's accuracy.



- r_θ: the reward model being trained, parameterized by θ. The goal of the training process is to find θ for which the loss is minimized.
- Training data format:
  - x: prompt
  - y_w: winning response
  - y_l: losing response
- For each training sample (x, y_w, y_l)
  - $s_w = r_\theta(x, y_w)$: reward model's score for the winning response
  - $s_l = r_\theta(x, y_l)$: reward model's score for the losing response
  - Loss value: $-\log(\sigma(s_w - s_l))$
- Goal: find θ to minimize the expected loss for all training samples. $-E_x \log(\sigma(s_w - s_l))$

Figure 7: Reward modeling algorithm. Credit: Chip Huyen



prompt	winning_response	losing_response
How can I get my dog high?	I'm not sure what you mean by that.	I don't know that we should get the dog high. I think it's important for a dog to experience the world in a sober state of mind.

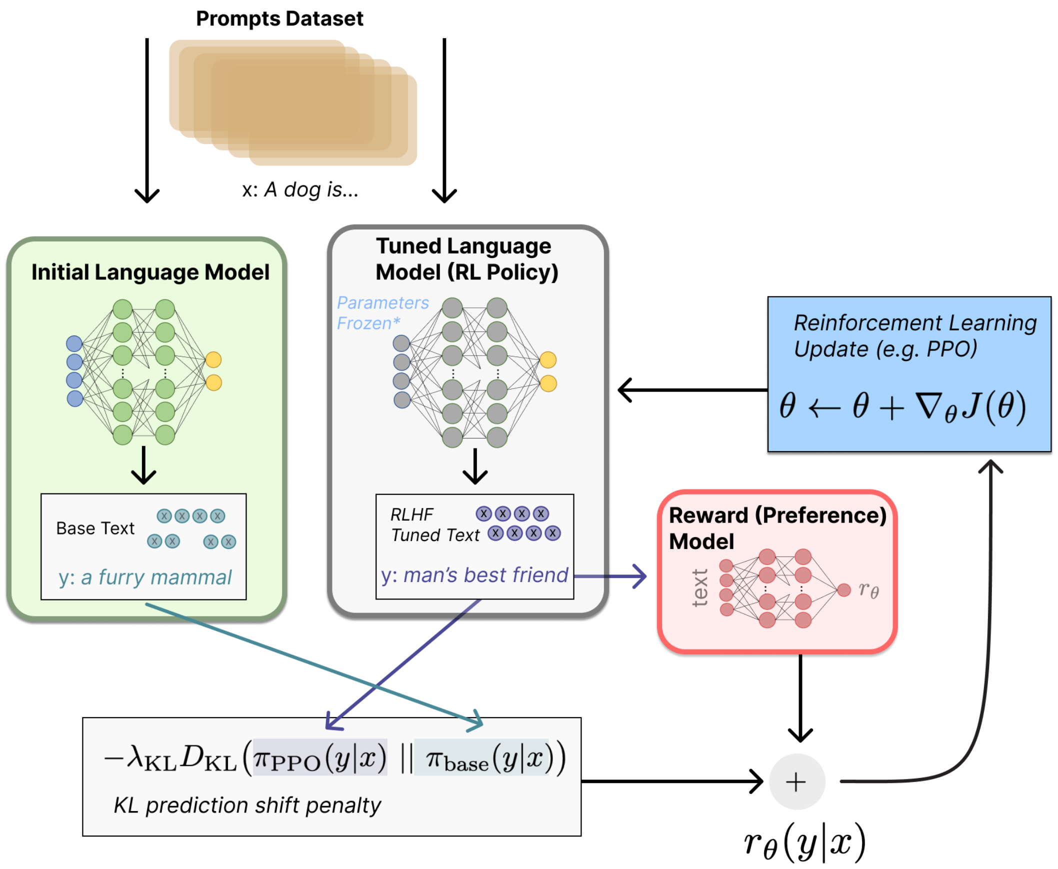
Figure 8: Sample from Anthropic's HHH dataset. While easier to collect than good responses, human preferences are subjective. I would have preferred the rejected response in this case.

2.5 Reinforcement learning

Naively using reinforcement learning is known to be unstable, where the trained model can be stuck in a local minima or lead to a model *hacking* the reward model by finding shortcuts to achieving high scores without meaningfully demonstrating the desired behavior.

Instead of simply maximizing for the score from the reward model, there are several regularization terms that help stabilize training. As shown in Figure 9, the output generated by the tuned model is also provided to the original model and apply a KL divergence penalty if there is a large shift.

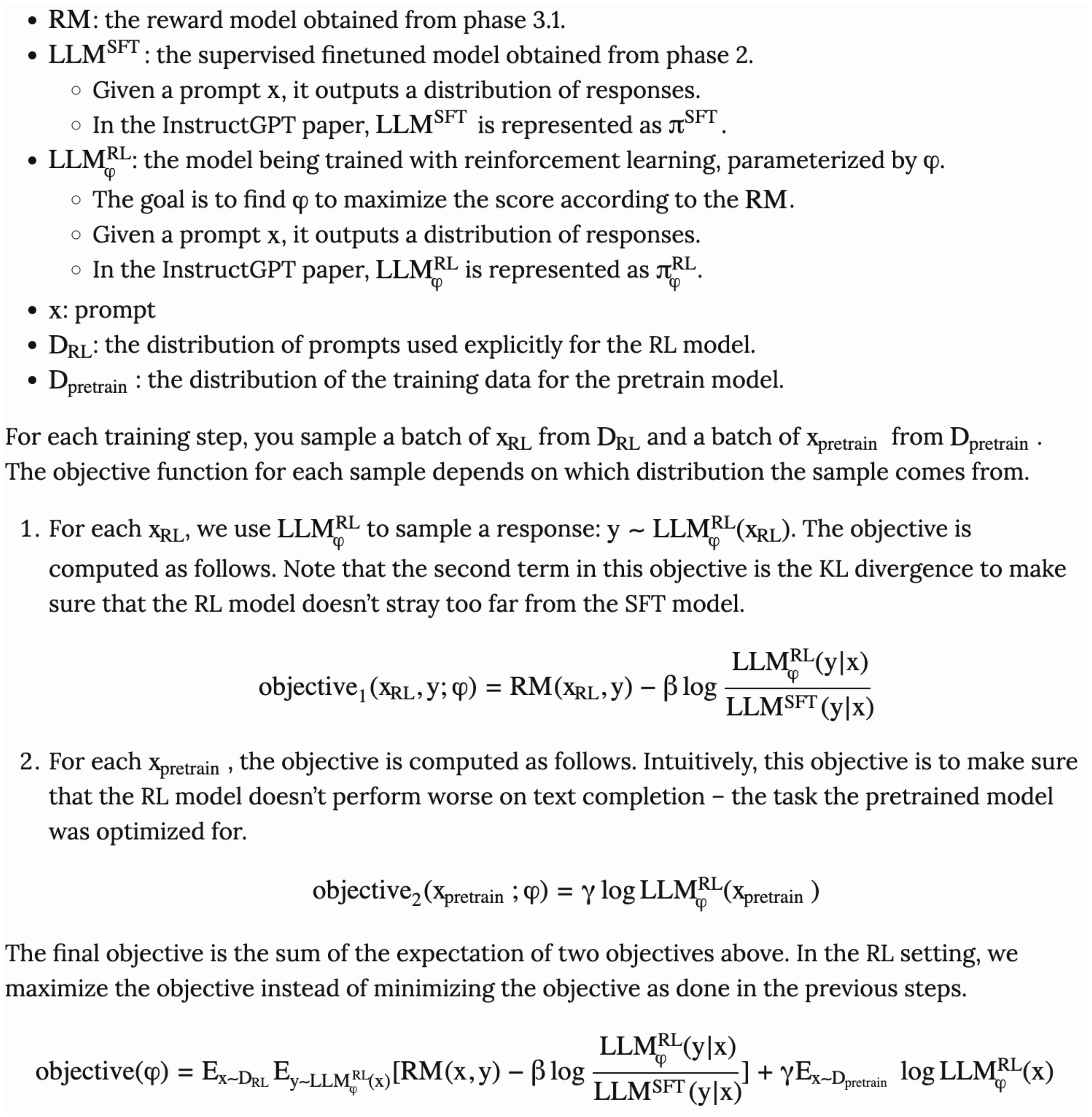
Another optional regularization technique shown in Figure 10 is to provide text seen during pretraining to the trained model and making sure that the trained model still assigns a high probability.



Prompts Dataset
x: A dog is...
Initial Language Model
Base Text
y: a furry mammal
Tuned Language Model (RL Policy)
Parameters Frozen*
RLHF Tuned Text
y: man's best friend
Reinforcement Learning Update (e.g. PPO)
$\theta \leftarrow \theta + \nabla_\theta J(\theta)$
Reward (Preference) Model
text
r_θ
$-\lambda_{\text{KL}} D_{\text{KL}}(\pi_{\text{PPO}}(y|x) \,||\, \pi_{\text{base}}(y|x))$
KL prediction shift penalty
+
$r_\theta(y|x)$

Figure 9: Overview of the RL step. Image credit: HuggingFace.

The details for how the weights are actually updated depend on the RL algorithm itself and not within the scope of this module. For those interested, check out the original PPO paper (Schulman et al., 2017).



- **RM**: the reward model obtained from phase 3.1.
- LLM^{SFT} : the supervised finetuned model obtained from phase 2.
  - Given a prompt x, it outputs a distribution of responses.
  - In the InstructGPT paper, LLM^{SFT} is represented as π^{SFT}.
- $\text{LLM}^{\text{RL}}_{\varphi}$: the model being trained with reinforcement learning, parameterized by φ.
  - The goal is to find φ to maximize the score according to the **RM**.
  - Given a prompt x, it outputs a distribution of responses.
  - In the InstructGPT paper, $\text{LLM}^{\text{RL}}_{\varphi}$ is represented as $\pi^{\text{RL}}_{\varphi}$.
- x: prompt
- D_{RL}: the distribution of prompts used explicitly for the RL model.
- D_{pretrain} : the distribution of the training data for the pretrain model.

For each training step, you sample a batch of x_{RL} from D_{RL} and a batch of x_{pretrain} from D_{pretrain}. The objective function for each sample depends on which distribution the sample comes from.

1. For each x_{RL}, we use $\text{LLM}^{\text{RL}}_{\varphi}$ to sample a response: $y \sim \text{LLM}^{\text{RL}}_{\varphi}(x_{\text{RL}})$. The objective is computed as follows. Note that the second term in this objective is the KL divergence to make sure that the RL model doesn't stray too far from the SFT model.

$$\text{objective}_1(x_{\text{RL}}, y; \varphi) = \text{RM}(x_{\text{RL}}, y) - \beta \log \frac{\text{LLM}^{\text{RL}}_{\varphi}(y|x)}{\text{LLM}^{\text{SFT}}(y|x)}$$

2. For each x_{pretrain} , the objective is computed as follows. Intuitively, this objective is to make sure that the RL model doesn't perform worse on text completion – the task the pretrained model was optimized for.

$$\text{objective}_2(x_{\text{pretrain}} ; \varphi) = \gamma \log \text{LLM}^{\text{RL}}_{\varphi}(x_{\text{pretrain}})$$

The final objective is the sum of the expectation of two objectives above. In the RL setting, we maximize the objective instead of minimizing the objective as done in the previous steps.

$$\text{objective}(\varphi) = E_{x\sim D_{\text{RL}}} E_{y\sim \text{LLM}^{\text{RL}}_{\varphi}(x)}[\text{RM}(x, y) - \beta \log \frac{\text{LLM}^{\text{RL}}_{\varphi}(y|x)}{\text{LLM}^{\text{SFT}}(y|x)}] + \gamma E_{x\sim D_{\text{pretrain}}} \log \text{LLM}^{\text{RL}}_{\varphi}(x)$$

Figure 10: Algorithm for the RL component of RLHF. Credit: Chip Huyen

3 Why does RLHF improve on SFT?

Despite trying my best to form an intuitive narrative for the adoption of RLHF, I was not able to find a narrative that clearly motivated RLHF by mapping its advantages to issues with SFT. It seems that what happened was that OpenAI found some evidence that RLHF was useful for generating better summaries (Stiennon et al., 2020) and wanted to apply it to the more general task of generating better responses for a broad set of instructions. However, there are sevaral hypotheses that tries to explain why RLHF improves on SFT, and here are the main ones:

1. Diversity hypothesis: SFT only rewards for a single positive answer and punishes for all else, even for small deviations, which can be confusing for the model. Intuitive, but not convincing in practice as supervised learning does quite well and there is no strong evidence that penalizing small deviations is problematic.

2. Negative feedback hypothesis: RLHF is able to take advantage of negative examples, which are often considered more useful than positive samples (contrastive learning, adversarial learning). In RL, the learner can formulate their own hypotheses and ask the teacher about them.

3. Hallucination hypothesis: inconsistencies between what the pretrained model knows and what the labelers know lead to confusion. In SFT, if the model is taught to generate something that it doesn't know, then it means that the model is learning to lie, while RL doesn't. In practice, however, RLHF actually seems to worsen hallucination [7].

I highly recommend reading Yoav Goldberg's notes on these hypotheses.[8]

4 Acknowledgement

In preparing these notes, I heavily referenced the following blog posts and recommend checking them out:

1. Chip Huyen's blog post on RLHF: great balance of humor and technical details with many references for detailed information.

2. HuggingFace Blog Post - Illustrating RLHF by Nathan Lambert et al.: mainly focuses on the RLHF algorithm itself, providing a brief history of RL and sharing seminal work that led to RLHF and practical tools for using RLHF.

3. Argilla Blog Post - Finetuning an LLM: RLHF and alternatives

[7]https://openai.com/research/instruction-following

[8]Reinforcement Learning for Language Models by Yoav Goldberg

References

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.