CÔNG TY CỔ PHẦN VCCORP

2ND WEEK REPORT

LEADER: MR. NGÔ VĂN VĨ

Post Training (RFHT, SFT, ...)

Lê Văn Hậu

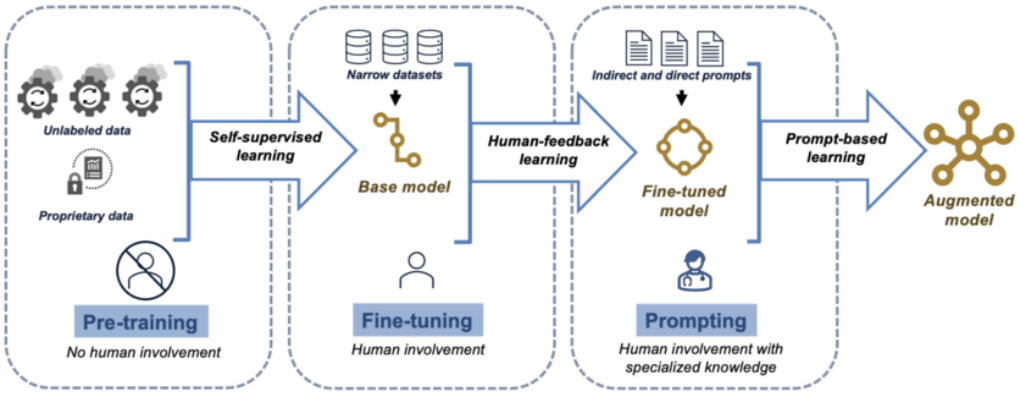
Contents

Second Week Report

FireFly

June 30, 2025

Post-training is a crucial phase in the development of Large Language Models (LLMs) that occurs after their initial pre-training on vast datasets. Its primary goal is to **align LLM behavior with human intent, ethical guidelines, and specific task requirements**, effectively refining the model to be more helpful, harmless, and honest. This phase addresses inherent limitations of pre-trained models, such as factual inaccuracies, logical inconsistencies, and the generation of undesirable or biased content.



General Benefits of Post-Training

The motivations for post-training are manifold:

- **Enhanced Reasoning and Factual Accuracy:** Post-training improves the model's ability to reason logically and generate factually correct information.

- **Alignment with User Intent and Ethics:** It fine-tunes the LLM to better understand and fulfill user instructions, while also adhering to ethical considerations and reducing harmful outputs.
- **Improved Task-Specific Performance:** Models can be adapted to excel in particular applications like question answering, summarization, or sentiment analysis.
- **Reduced Undesirable Outputs:** It helps mitigate biases, hallucinations, and the generation of toxic or unsafe content.

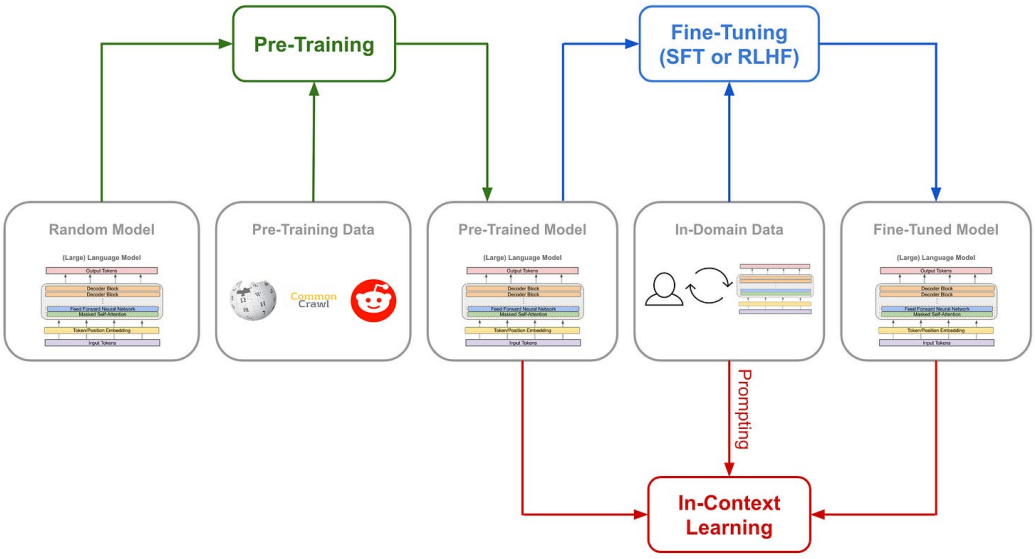
However, it's important to note that post-training, particularly fine-tuning, can sometimes lead to **catastrophic forgetting** (where the model forgets previously learned information) or even **safety degradation** if not carefully managed.

Key Post-Training Methodologies

Three prominent methodologies dominate the post-training landscape: Supervised Fine-Tuning (SFT), Reinforcement Learning from Human Feedback (RLHF), and Direct Preference Optimization (DPO).

—

1 Supervised Fine-Tuning (SFT)



Core Principles:

SFT involves further training a pre-trained LLM on a dataset of high-quality **instruction-following or dialogue examples**. The model learns to map specific inputs (instructions) to desired outputs by minimizing the **negative log-likelihood** of the correct output tokens. Essentially, it's about teaching the model to follow commands based on curated examples.

Algorithms and Implementation:

The core algorithm for SFT is based on **cross-entropy loss**. During training, the model's generated sequence for a given input is compared to the ground truth (desired) sequence, and the loss is calculated. Optimizers like **AdamW** are commonly used to adjust the model's weights. Implementation typically involves:

1. **Data Collection:** Gathering or creating high-quality instruction-response pairs.

2. **Data Formatting:** Structuring the data into prompts and desired completions.

3. **Model Fine-tuning:** Training the pre-trained LLM on this dataset using standard supervised learning techniques.

Trade-offs and Characteristics:

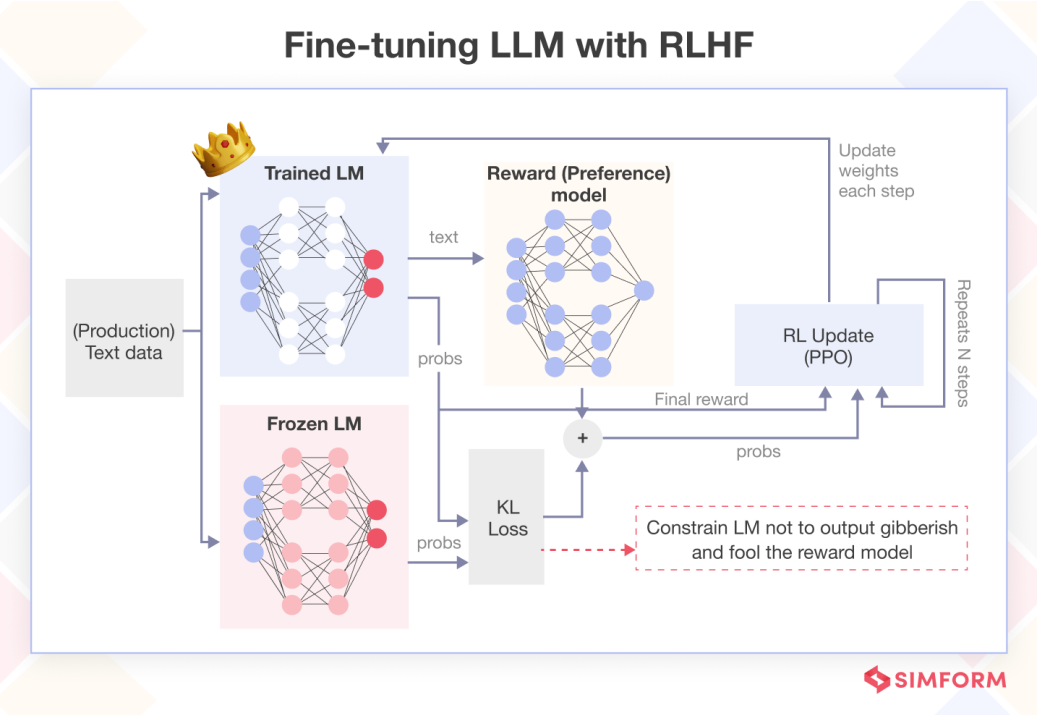
- **Advantages:** Relatively **simple to implement**, computationally less intensive than RLHF, and effective for directly adapting the model to specific tasks or instruction formats.
- **Disadvantages:** Can lead to **limited diversity** in responses, amplify **popularity bias** present in the training data, and is susceptible to **overfitting** on the fine-tuning dataset, potentially leading to unintended behaviors like increased hallucination if the data is not meticulously curated.
- **Data Requirements:** Requires a dataset of high-quality, diverse instruction-response examples.
- **Computational Cost:** Significant for large models, but generally lower than RLHF.

Challenges and Limitations:

A major challenge is ensuring the **quality and diversity** of the SFT dataset. Poor data can embed biases or lead to the model only excelling at the exact types of instructions it was trained on, lacking generalization.

—

2 Reinforcement Learning from Human Feedback (RLHF)



Core Principles:

RLHF aims to align LLMs with complex human preferences that are difficult to capture with simple rules or supervised data. It achieves this by training a separate **reward model** to predict human preferences, and then using this reward model to fine-tune the LLM through reinforcement learning.

Algorithms and Implementation:

RLHF typically follows a four-step process:

1. **Pre-training:** The initial LLM is pre-trained on a broad text corpus.
2. **Human Preference Data Collection:** Human annotators rank or compare different outputs generated by the LLM for a given prompt, indicating which one is preferred.

3. **Reward Model (RM) Training:** A separate model (the RM) is trained on this human preference data. Its goal is to predict the reward (preference score) of any given LLM output, effectively learning to mimic human judgments.

4. **Reinforcement Learning Fine-tuning:** The original LLM is then fine-tuned using a reinforcement learning algorithm (e.g., **Proximal Policy Optimization - PPO**). The RM's predicted reward guides the LLM's learning process, encouraging it to generate responses that are highly rated by the RM, and by extension, by humans.

Trade-offs and Characteristics:

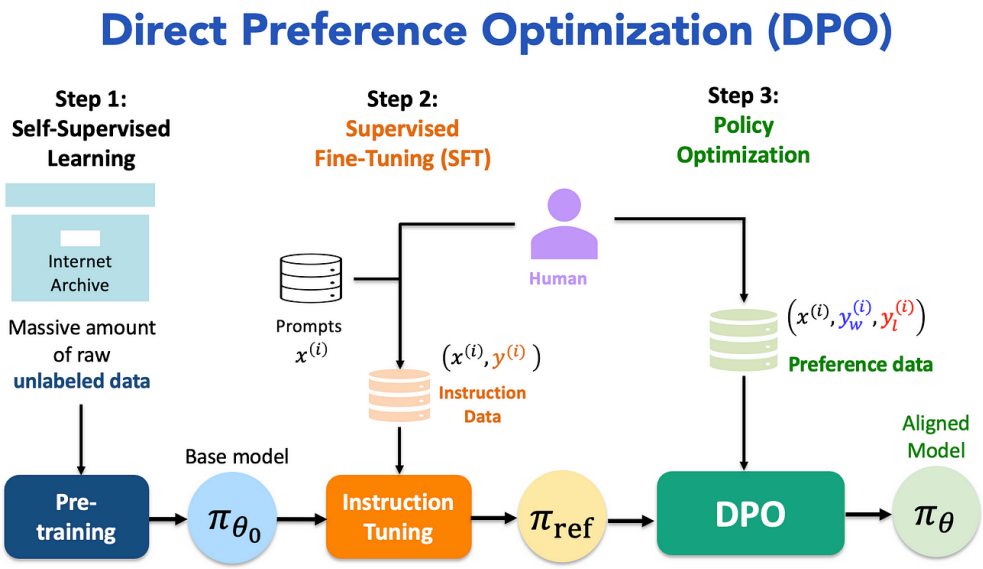
- **Advantages:** Enables **detailed customization** and alignment with nuanced human preferences, handles complex and abstract goals effectively, and has demonstrated significant success in improving model behavior.
- **Disadvantages: Highly complex** to implement, computationally very expensive due to the multi-stage process and online sampling, heavily relies on the quality and variety of human feedback, and is susceptible to **reward hacking** (where the model exploits flaws in the reward function without truly aligning with human intent) and **mode collapse**.
- **Data Requirements:** Requires extensive human preference data (rankings, comparisons).
- **Computational Cost:** Very high due to the iterative nature of RL and the training of both the policy and reward models.

Challenges and Limitations:

Challenges abound in RLHF, especially concerning **human feedback quality** (biases, sycophancy, data poisoning), **reward model misspecification** (difficulty in representing diverse human values), and **policy optimization difficulties** (distribution shifts, adversarial exploitability, mode collapse). Best practices involve multi-layered safety approaches, careful feedback design, and robust RL algorithms.

—

3 Direct Preference Optimization (DPO)



Core Principles:

DPO offers a simpler and more stable alternative to RLHF. Instead of explicitly training a reward model, DPO **directly optimizes the LLM policy using human preference data**. It reframes the RL problem as a simple classification problem, enabling the model to learn preferences without the intermediate reward model.

Algorithms and Implementation:

DPO directly optimizes the policy by leveraging the relationship between the optimal policy and the optimal reward function in the RL context. Its core is a **binary cross-entropy loss function**. For a given pair of preferred (y_w) and dispreferred (y_l) responses to a prompt (x), the DPO loss maximizes the probability of the preferred response relative to the dispreferred response, while implicitly satisfying a KL-divergence constraint with the initial (SFT-tuned) policy. The loss is effectively:

$$L_{DPO}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

where π_θ is the current policy, π_{ref} is the reference policy (typically the SFT model), and β is a hyperparameter controlling the strength of the KL divergence penalty.

Implementation steps are:

1. **Initial SFT:** An LLM is typically first fine-tuned using SFT.
2. **Preference Data Collection:** Pairs of preferred and dispreferred responses for various prompts are collected, similar to RLHF.
3. **Direct Optimization:** The SFT-tuned model is then further trained directly using the DPO loss function on these preference pairs.

Trade-offs and Characteristics:

- **Advantages:** Significantly **simpler and more stable** than RLHF (bypassing the need for a separate reward model and complex RL algorithms), **faster adjustments**, and **lower computational demand**. It provides more direct alignment with human preferences.
- **Disadvantages:** May not perform as well with highly **nuanced feedback** compared to RLHF (which can model continuous reward signals). Data collection can still be labor-intensive. Can be susceptible to **biased solutions** and performance degradation due to **distribution shifts** between the SFT and DPO data.
- **Data Requirements:** Pairs of preferred and dispreferred responses.
- **Computational Cost:** Significantly lower than RLHF, making it more scalable.

Challenges and Limitations:

DPO can struggle with very **challenging tasks** (e.g., complex code generation) where a strong reward signal is critical. It is also sensitive to **noisy or controversial data** and might find biased solutions if not carefully implemented. Optimal selection of the β hyperparameter is crucial for performance.

Frameworks, Tools, and Libraries

Several frameworks and tools support these post-training methodologies, making them more accessible to developers:

- **Hugging Face TRL (Transformer Reinforcement Learning):** A leading library that provides implementations for SFT, RLHF (including PPO and GPO), and DPO, making it a go-to for fine-tuning LLMs with human feedback.
- **OpenAI:** While not providing open-source libraries in the same way as Hugging Face, OpenAI's API offers options for "preference fine-tuning" and "reinforcement fine-tuning," which are based on methods like DPO and RLHF, respectively.
- **DeepMind:** Has extensively researched and implemented these techniques internally, with projects like Reinforced Self-Training (ReST) emerging as alternatives or complements to RLHF.

In summary, post-training is essential for transforming general-purpose pre-trained LLMs into specialized, aligned, and safe assistants. The choice between SFT, RLHF, and DPO depends on factors such as the complexity of the desired behavior, available computational resources, and the nature of the human feedback data.