

RL without Tears:

Chapter 4: Improved RL for LLMs

Chenglong Wang
Northeastern University, China

WANGCHENGLONG@MAIL.NEU.EDU.CN

Hang Zhou
Northeastern University, China

STCEUM@GMAIL.COM

Tongran Liu
Institute of Psychology, Chinese Academy of Sciences, China

LIUTR@PSYCH.AC.CN

Jingbo Zhu
Northeastern University, China

ZHUJINGBO@MAIL.NEU.EDU.CN

Tong Xiao
Northeastern University, China

XIAOTONG@MAIL.NEU.EDU.CN

4 Improved RL for LLMs

In the previous section, we introduced some improvements to RL, such as importance sampling and reward baseline techniques. However, directly applying them to train LLMs still presents numerous challenges. In this section, we will delve deeper into the improvements for using RL to train LLMs.

4.1 Advanced Reward Models

Reward models are a fundamental component in RL, as they define what a policy model optimizes for. Therefore, the quality of reward model training directly influences the effectiveness of LLM optimization during RLHF. A poorly trained reward model can lead to a suboptimal LLM, while a well-optimized reward model ensures that the LLM is effectively aligned with the desired behaviors and objectives. Here we will introduce several methods for obtaining advanced reward models. Our discussion will be relatively general, since reward models are widely used in many RL problems, such as robot planning and control. This broad applicability makes it straightforward to adapt the methods discussed here to other related applications.

4.1.1 Automatic Preference Data Generation

Although learning from human preferences is an effective and popular method for aligning LLMs, annotating preference data is costly. Relying on human feedback not only faces scalability limitations but may also introduce bias, as human feedback is inherently subjective. Consequently, AI-based feedback methods offer a promising solution to address these scalability and consistency issues, avoiding the limitations associated with human annotators.

One simple method is to generate preference data using an LLM. Given a set of inputs, we first use an LLM to generate pairs of outputs. Then, we prompt the LLM to label the preference between each pair of outputs, along with its corresponding input. Below is an example of prompting the LLM to generate a preference label for a pair of outputs.

Consider a homework assistant scenario in which a student types an input. You will review two outputs to this input. Please indicate which output you prefer. A good output should be informative, accurate, and well-articulated. It should directly address the student’s question, provide thorough explanations or solutions, and maintain an encouraging tone.

Input:

Give me three tips to improve my accuracy in solving math problems.

Output A:

Improving math accuracy requires careful attention to detail, strategic problem-solving techniques, and consistent practice. Here are three tips:

1. Practice regularly:

- Consistent practice builds fluency and familiarity with different problem types...

...

By combining your advice with these additional tips, anyone can significantly enhance their math problem-solving abilities.

Output B:

There are three tips for improving accuracy in solving math problems:

1. Practice regularly.

2. Understand the concepts.

3. Double-check your work.

Output A is preferred.

Once we collect such preference labels, we can use them, along with the output pair and input, to train the reward model. Of course, these labels are not entirely accurate either. Recent studies have shown that AI-based feedback often exhibits a position bias problem, making it more likely to prefer the output shown first (Zheng et al., 2023). We can consider demonstrating a few examples or using advanced prompting techniques, such as Chain-of-Thought (CoT), to improve the labeling performance (Liu et al., 2023).

For data generation, although it is easy to scale up, it is often necessary to ensure the data is accurate and diverse. Here, the data quality and diversity issues involve not only the labeling of preferences but also the inputs and outputs of the model. Therefore, we often need to use a variety of techniques to obtain large-scale, high-quality data. For example, one can generate diverse model outputs and annotations by using different LLMs, prompts, in-context demonstrations, and so on (Cui et al., 2024b). Furthermore, Dubois et al. (2023) report that the variability in pairwise preference data is important for training LLMs from either human or AI feedback.

While learning from AI feedback is highly scalable and generally objective, this method is more suited to well-defined tasks where objective performance metrics are available. By contrast, learning from human feedback is more advantageous when aligning AI systems with human values, preferences, and complex real-world tasks that require an understanding of subtle or subjective context. These methods can be combined to train LLMs that benefit from both human insights and the scalability of AI feedback.

4.1.2 Reward Shaping

As discussed in Section ??, while the reward model is effective in capturing human preferences, it provides sparse rewards. These rewards, known as delayed rewards, are received only at the end of the output generation process, as opposed to intermediate rewards that would be distributed continuously throughout it.

x Give me three tips to improve my accuracy in solving math problems.

There are three tips for improving accuracy in solving math problems:

y 1. Practice regularly.

Intermediate Reward: +1

2. Understand the concepts.

Delayed Reward: +10.6

3. Double-check your work.

Figure 1: An example of reward shaping. Here, length-based shaping rewards are implemented, e.g., awarding a +1 intermediate reward for every 10 tokens generated. Additionally, a delayed reward from the reward model evaluates the overall quality of the output at the end.

In fact, dealing with sparse rewards has long been a concern in RL, and has been one of the challenges in many practical applications. For example, robotics often needs to shape the reward function to ease optimization rather than relying solely on end-of-sequence rewards. Various methods have been developed to address this issue. One common approach is reward shaping, where the original function is modified to include intermediate rewards, thereby providing more immediate feedback. Here, the intermediate reward is often an indirect signal for improving the delayed reward. For example, as shown in Figure 1, setting a length-based reward as an intermediate reward encourages the generation of more content, potentially increasing the overall quality of the final output as assessed by the delayed reward from the reward model. Additional examples of reward shaping in training LLMs can be found in [Kumar et al. \(2025\)](#). In this way, we can obtain

$$r'_t = r_t + f(\mathbf{x}, \mathbf{y}_{<t}, y_t) \quad (1)$$

where $r'(\cdot)$ is the transformed reward, $r(\cdot)$ is the original delayed reward function, and $f(\cdot)$ is the shaping reward function that provides an intermediate reward. To ensure the optimality of the policy under the transformed reward function, the shaping reward function can be given in the form

$$f(\mathbf{x}, \mathbf{y}_{<t}, y_t) = \gamma \Phi(\mathbf{x}, \mathbf{y}_{<t+1}, y_{t+1}) - \Phi(\mathbf{x}, \mathbf{y}_{<t}, y_t) \quad (2)$$

where $\Phi(\cdot)$ is called the potential value function. If we define $\Phi(\cdot)$ as the common value function and substitute Eq. (2) into Eq. (1), we obtain

$$r'_t = r_t + \gamma V_{t+1} - V_t \quad (3)$$

It is interesting to see that this function is exactly the same as the advantage function used in PPO. This relates advantage-based methods to reward shaping: the advantage is essentially a shaped reward.

In practice, an alternative method for implementing reward shaping in training LLMs involves utilizing the reward model to provide intermediate rewards based on the differences between output rewards ([Bahdanau et al., 2017](#); [Wu et al., 2023](#)). For example, at time step t , the intermediate reward r_t can be obtained by

$$r_t = R(\mathbf{x}, \mathbf{y}_{<t+1}) - R(\mathbf{x}, \mathbf{y}_{<t}) \quad (4)$$

However, in this case, it is crucial that the reward model is capable of accurately assigning rewards for partial outputs, i.e., $\{\mathbf{y}_{<2}, \mathbf{y}_{<3}, \dots, \mathbf{y}_{<T}\}$.

In addition to reward shaping, another method to address the sparse reward issue is adopting curriculum learning, where tasks are structured sequentially with gradually increasing complexity. This can help models master simpler tasks first, which prepares them for more complex challenges as their skills develop. There

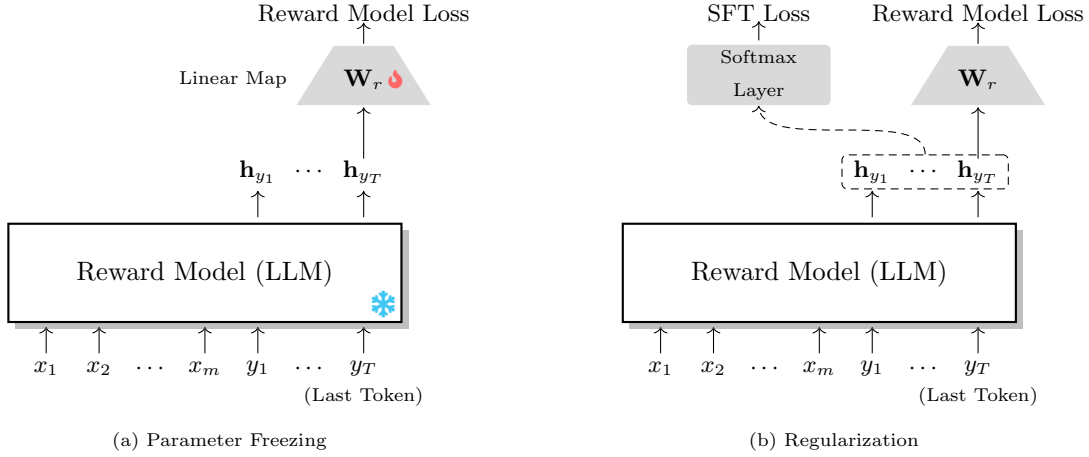


Figure 2: When training reward models, we can use parameter freezing and regularization methods to preserve the features of the LLM and thereby improve reward generalization.

are many methods that can mitigate the impact of sparse rewards, such as Monte Carlo methods and intrinsic motivation. Most of these methods are general, and the discussion of them can be found in the broader literature on RL, such as [Sutton & Barto \(2018\)](#)’s book.

4.1.3 Improved Reward Generalization

As discussed in Section ??, reward models are trained using preference data and subsequently used to optimize LLMs. However, a problem arises: the data distribution during LLM optimization may differ from the distribution of the preference data, making it challenging for the reward model to generalize to unseen input-output pairs.

A well-known failure mode associated with this problem is commonly referred to as *overoptimization* or *reward hacking*, where the optimization stage improves the reward model score but deteriorates the alignment with true rewards ([Gao et al., 2023](#); [Eisenstein et al., 2023](#)). This mode occurs because the reward model may incorrectly assign high rewards to unseen input-output pairs, leading the LLM to learn and optimize for behaviors that do not truly align with the desired behaviors and objectives. For example, consider a scenario from Section ?? where the reward model is designed to favor informative and accurate outputs for a homework assistant. However, if the reward model fails to generalize to an overly verbose output and assigns a high reward to it (perhaps due to its length or certain keywords), the LLM may learn to prioritize generating a long output, which is not actually more informative but receives a higher reward according to the reward model. Consequently, the LLM becomes misaligned with its true objectives, delivering concise and relevant information, because it optimizes for the incorrect reward signal from the reward model with weak generalization.

Addressing this generalization problem is challenging, and no mature solution exists yet. The ideal approach would be to develop an oracle reward model that perfectly captures the true objectives of the task and generalizes across all input-output pairs during LLM optimization. However, creating such a model is extremely difficult due to the complexity of the real-world environment, as well as the challenge of collecting sufficient preference data. Instead, a more practical approach is to combine multiple reward models, improving generalization and providing more accurate rewards ([Coste et al., 2024](#)).

Given a set of reward models, combining them is straightforward, and in some cases, we can simply treat this problem as an ensemble learning problem. A simple yet common approach is to average the outputs of

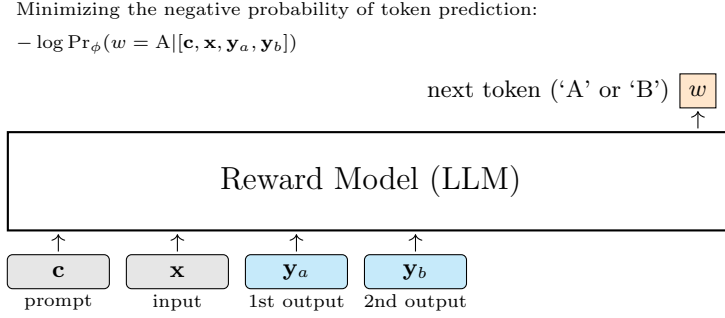


Figure 3: In generative reward models, we use an LLM to predict the label token given a prompt, an input, and a pair of outputs. This model can be trained in the same way as standard LLMs.

these models to obtain a more precise reward estimation:

$$R_{\text{combine}}(\mathbf{x}, \mathbf{y}) = \frac{1}{K} \sum_{k=1}^K w_k \cdot R_k(\mathbf{x}, \mathbf{y}) \quad (5)$$

where $R_k(\cdot)$ is the k -th reward model in the ensemble, w_k is the weight of $R_k(\cdot)$, and K is the number of reward models. This combined reward can then be used to supervise the training of a policy. In fact, there are many ways to combine different models; for example, one can make predictions using Bayesian model averaging or develop a fusion network to learn to combine the predictions from different models. Alternatively, one can frame this task as a multi-objective optimization problem and use multiple reward models to train the policy simultaneously. These methods have been intensively discussed in the literature on optimization and machine learning (Miettinen, 1999; Bishop, 2006).

On the other hand, to improve the generalization of reward models, it is important to prevent overfitting to preference data. Reward models are typically trained on an SFT LLM, which has a strong generalization capability. However, training the LLM with a large amount of preference data could lead to overfitting, which may ultimately reduce generalization performance. A common strategy to mitigate this issue is to employ a parameter freezing technique, which helps preserve the original features of the LLM while learning the reward model. Another practical approach is to add a regularization term to the preference learning loss function, which helps regulate the features of the LLM (Yang et al., 2024). For example, we can use a simple SFT loss as regularization by adding a term that maximizes the probability of the preferred output $\mathbf{y}_a$ to Eq. (??):

$$\mathcal{L}_{\text{reg}}(\phi) = -\mathbb{E}_{(\mathbf{x}, \mathbf{y}_a, \mathbf{y}_b) \sim \mathcal{D}_r} [\log \Pr_{\phi}(\mathbf{y}_a \succ \mathbf{y}_b | \mathbf{x}) + \alpha \log(\Pr_{\theta}(\mathbf{y}_a | \mathbf{x}))] \quad (6)$$

where α is a balancing factor. We further illustrate the parameter freezing and regularization methods in Figure 2. Note that the regularization term applies only to the LLM. That is, when optimizing this term, only the parameters of the LLM are updated, while the parameters of the reward linear map remain fixed. Therefore, in this equation, the parameter associated with the regularization term is θ , which specifically refers to the parameters of the LLM.

4.1.4 Generative Reward Models

Reward models are typically trained as discriminative models to assign numerical rewards to outputs and classify them as preferred or dispreferred. However, this method does not leverage the text-generation capabilities for which LLMs are fundamentally designed (Zhang et al., 2025b; Wang et al., 2025). For example, the discriminative reward model cannot perform CoT reasoning. To address this, an LLM can alternatively be employed as a reward model, thus endowing it with the ability to engage in text generation

and reasoning, as depicted in Figure 3. This model works as follows. First, we input a prompt $\mathbf{c}$, along with the tuple $(\mathbf{x}, \mathbf{y}_a, \mathbf{y}_b)$, to the LLM. The prompt is a description of the task, as demonstrated in the example below.

You are given two outputs to an input. Evaluate which output is better based on quality, relevance, and clarity. If the first output is better, return ‘A’. If the second output is better, return ‘B’.

Then, the LLM predicts subsequent tokens based on this input sequence. Let w be the label token predicted by the LLM and w^* be the annotated preference label. If $w^* = A$, it indicates a preference for $\mathbf{y}_a$ over $\mathbf{y}_b$; if $w^* = B$, then $\mathbf{y}_b$ is preferred. Note that here $\mathbf{y}_a$ and $\mathbf{y}_b$ do not have a pre-defined preference relationship as described in Section ???. Their relationship is instead represented by the label token.

The loss function can be defined as the log-probability of predicting the annotated preference label:

$$\mathcal{L}_{\text{gen}}(\theta) = -\mathbb{E}_{(\mathbf{c}, \mathbf{x}, \mathbf{y}_a, \mathbf{y}_b, w^*) \sim \mathcal{D}_r} [\log \Pr_{\theta}(w = w^* | \mathbf{s})] \quad (7)$$

where $\mathbf{s}$ denotes the string $[\mathbf{c}, \mathbf{x}, \mathbf{y}_a, \mathbf{y}_b]^1$, and $\Pr_{\theta}(\cdot)$ denotes the probability of token prediction by the LLM.

Although we discuss methods for training a generative reward model here, an interesting question arises: how can we use the generative reward model in RLHF? We provide some guidance as follows. Specifically, when applying this generative reward model to provide a reward for an input-output pair $(\mathbf{x}', \mathbf{y}')$, we can generate a reference output $\mathbf{y}_{\text{ref}}$ by using the LLM, for example, through greedy search, and concatenate $\mathbf{x}'$, $\mathbf{y}'$ and $\mathbf{y}_{\text{ref}}$ into $\mathbf{s}' = [\mathbf{c}, \mathbf{x}', \mathbf{y}', \mathbf{y}_{\text{ref}}]$. Additionally, to mitigate the positional bias problem (Wang et al., 2024b), we can introduce an alternative input order by transposing the positions of output, i.e., presenting $\mathbf{y}_{\text{ref}}$ before $\mathbf{y}'$, to construct a secondary input string $\mathbf{s}'_T = [\mathbf{c}, \mathbf{x}', \mathbf{y}_{\text{ref}}, \mathbf{y}']$. The reward for $(\mathbf{x}', \mathbf{y}')$ is thus defined as the probability that $\mathbf{y}'$ is preferred over $\mathbf{y}_{\text{ref}}$:

$$R_{\phi}(\mathbf{x}', \mathbf{y}') = \frac{\Pr_{\theta}(w = A | \mathbf{s}') + \Pr_{\theta}(w = B | \mathbf{s}'_T)}{2} \quad (8)$$

where the reward ranges from 0 to 1.

To further improve the generative reward model, we can label the relevant explanation to enable the generative reward model to produce a CoT rationale (Zhang et al., 2025a; Wang et al., 2026b). In this case, we use a prompt $\mathbf{c}$ with a CoT rationale generation instruction, as shown below.

You are given two outputs to a user input. Evaluate which output is better based on quality, relevance, and clarity. If the first output is better, return ‘A’. If the second output is better, return ‘B’. Note that, before presenting the evaluation results, you should provide a detailed analytical process and reasoning steps.

We can then define a new loss function for training the generative reward model as follows:

$$\mathcal{L}_{\text{gen}}(\theta) = -\mathbb{E}_{(\mathbf{c}, \mathbf{x}, \mathbf{y}_a, \mathbf{y}_b, \mathbf{rat}, w^*) \sim \mathcal{D}_r} [\log \Pr_{\theta}(\mathbf{rat} | \mathbf{s}) + \log \Pr_{\theta}(w = w^* | [\mathbf{s}, \mathbf{rat}])] \quad (9)$$

where $\mathbf{rat}$ is the labeled CoT rationale for generating a preference between $\mathbf{y}_a$ and $\mathbf{y}_b$. In practice, we can obtain the evaluation results by prompting a standard LLM (also known as LLM-as-a-judge), as demonstrated in Section 4.1.1. However, this approach typically underperforms compared to LLMs trained with preference data (Zhang et al., 2025a; Mahan et al., 2024). One reason is that standard LLMs are not fine-tuned specifically for the reward modeling task, and as a result, they may not fully capture the nuanced

¹In this work, we will use $\mathbf{s}$ interchangeably to refer to either the tuple of a training sample or a string representing that tuple.

decision-making process that aligns better with human preferences. Also, LLMs trained with preference data learn to prioritize outputs based on feedback, enhancing their capability to evaluate outputs according to the desired behaviors and objectives.

Although incorporating CoT rationales into generative reward models improves preference learning, it also introduces a probability degeneration issue in Eq. (8). Specifically, preference token probabilities often become saturated near 0 or 1, resulting in limited reward variance. This issue is particularly challenging for RL algorithms such as GRPO, where relative reward differences are essential for policy optimization. To address this limitation, researchers investigate ranking-based reward construction approaches. The basic idea is to use only the ranking capability of generative reward models to derive rewards for RL (Wang et al., 2026c). Interested readers can refer to this work for further details.

4.1.5 Rubric-based Reward Models

While generative reward models have shown strong performance in preference prediction, they typically produce an overall judgment for each response or response pair. For complex open-ended tasks, such holistic evaluation may be insufficient because response quality often depends on multiple instruction-specific aspects. To address this issue, we can employ rubric-based reward modeling. The basic idea is to introduce a set of task-specific criteria that explicitly guide the reward model on *what aspects to evaluate* when predicting preferences. In practice, these criteria can be organized into different rubric formats, as illustrated below, where the underlined text in each example represents the rubric.

- **Holistic Rubric.** A holistic rubric provides multiple evaluation criteria but asks the reward model to produce a single overall judgment. For example,

Reward Model Input:

Input: Give me three tips to improve my accuracy in solving math problems.

Output A: Improving math accuracy requires careful attention to detail and consistent practice. Here are three tips: practice regularly, understand the concepts, and ...

Output B: Practice more, read carefully, and check your answers.

Please compare the two outputs based on accuracy, informativeness, clarity, and helpfulness, and give an overall preference.

Reward Model Output:

Output A is better overall because it provides more informative and helpful suggestions. ... Therefore, the reward model assigns a higher reward to Output A and selects Output A as the preferred response.

- **Criterion-wise Scoring Rubric.** A criterion-wise scoring rubric asks the reward model to assign a separate score to each evaluation criterion. These criterion-level scores are then averaged or weighted to obtain the final reward.

Reward Model Input:

Input: Give me three tips to improve my accuracy in solving math problems.

Output A: Improving math accuracy requires careful attention to detail and consistent practice. Here are three tips: practice regularly, understand the concepts, and ...

Output B: Practice more, read carefully, and check your answers.

Please score each output from 1 to 5 on the following criteria: accuracy, informativeness, clarity, and helpfulness.

Reward Model Output:

Output A: Accuracy 5/5; Informativeness 5/5; Clarity 4/5; Helpfulness 5/5.

Output B: Accuracy 4/5; Informativeness 2/5; Clarity 4/5; Helpfulness 3/5. ... After aggregating the criterion-level scores, Output A receives the higher overall score and is preferred over Output B.

- **Checklist Rubric.** A checklist rubric decomposes evaluation into atomic and verifiable conditions. Each condition is evaluated independently, typically using binary labels such as *Pass* or *Fail*.

Reward Model Input:

Input: Give me three tips to improve my accuracy in solving math problems.

Output A: Improving math accuracy requires careful attention to detail and consistent practice. Here are three tips: practice regularly, understand the concepts, and ...

Output B: Practice more, read carefully, and check your answers.

Please check whether each output satisfies the following requirements:

1. Does the response provide exactly three tips?
2. Are the tips relevant to improving math accuracy?
3. Does the response explain the suggested tips?
4. Is the response clear and easy to understand?

Reward Model Output:

Output A: Pass, Pass, Pass, Pass.

Output B: Pass, Pass, Fail, Pass. ... Since Output A satisfies all checklist items while Output B does not explain the tips, Output A is assigned the higher reward.

- **Hierarchical Rubric.** A hierarchical rubric organizes evaluation criteria into multiple levels. The reward model first evaluates candidate outputs using lower-level criteria and then aggregates these results into higher-level dimensions to obtain the final reward.

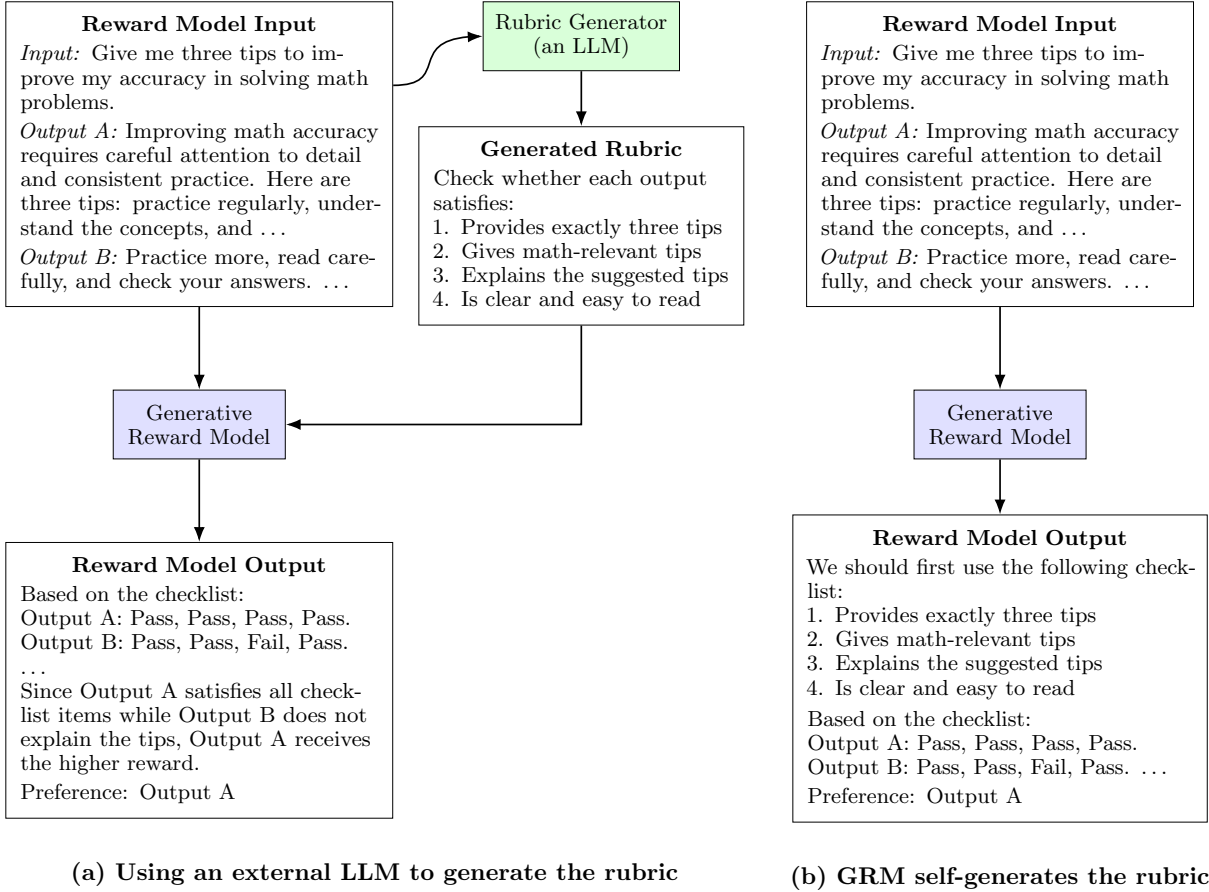


Figure 4: Two commonly used approaches for automatic rubric generation. We use checklist rubrics as an illustrative example.

Reward Model Input:

Input: Give me three tips to improve my accuracy in solving math problems.

Output A: Improving math accuracy requires careful attention to detail and consistent practice. Here are three tips: practice regularly, understand the concepts, and ...

Output B: Practice more, read carefully, and check your answers.

Please evaluate the outputs using the following hierarchical criteria:

Content Quality: Accuracy, Informativeness

Presentation Quality: Clarity, Helpfulness

Reward Model Output:

Output A: Content Quality 5/5; Presentation Quality 4.5/5.

Output B: Content Quality 3/5; Presentation Quality 3.5/5. ... By aggregating the lower-level evaluations into the two higher-level dimensions, Output A obtains the stronger final reward and is selected as the better response.

It is worth noting that reasoning can also be incorporated into rubric-based evaluation. For example, in a checklist-style rubric, the reward model can be asked to reason about each criterion before producing the corresponding Pass/Fail judgment. Additionally, although we use pairwise evaluation, i.e., taking $(\mathbf{x}, \mathbf{y}_a, \mathbf{y}_b)$

as reward model input, as the running example, rubric-based reward modeling is equally applicable to pointwise evaluation, i.e., taking $(\mathbf{x}, \mathbf{y})$ as reward model input.

The quality of the rubric is crucial to the performance of rubric-based reward models. As a result, recent work has devoted increasing attention to constructing high-quality rubrics for reward modeling. Similar to many other components in machine learning, rubric acquisition generally follows two approaches: manual design and automatic generation.

For manual design, one straightforward approach is to recruit human experts to write rubrics based on their domain knowledge and task requirements. However, it is often difficult to design a rubric that comprehensively covers the diverse cases that may arise across different inputs and tasks. This is because evaluation criteria that are appropriate for one case may be insufficient for another. Additionally, manually designed rubrics inherit the prompt sensitivity of LLM-based evaluation: even when the underlying evaluation dimensions remain the same, small differences in wording can lead to noticeably different results.

One promising strategy is to generate rubrics dynamically based on the input. As shown in Figure 4, existing approaches generally fall into two categories. The first uses an external LLM to generate task-specific rubrics from the input and evaluation requirements, which are then provided to the reward model for preference prediction. The second allows the generative reward model to generate its own rubrics for the current input and then evaluate candidate responses according to these self-generated criteria.

Both approaches formulate rubric generation as an explicit task. This naturally leads to another idea: can we optimize the rubric generation process itself to produce better rubrics? The answer is yes. Rubrics generated directly by LLMs may suffer from limited coverage, redundant criteria, or preference misalignment (Liu et al., 2026; Shen et al., 2026). To improve rubric quality, we can explicitly refine the generated criteria. For example, OpenRubrics generates rubrics by contrasting preferred and rejected responses to identify more discriminative rules and principles (Liu et al., 2026). We can further learn the rubric generation process itself through RL training (Xu et al., 2026). More recent studies go one step further by allowing rubrics to evolve with the policy, so that the evaluation criteria continue to capture new weaknesses as the model improves (Rezaei et al., 2025; Ding et al., 2026; Yu et al., 2026). Rubric optimization is becoming an active research direction, and interested readers can refer to the aforementioned works for further details.

4.1.6 Reward Model Evaluation

After training a reward model, an important question is *how to evaluate whether the learned reward function can accurately capture human preferences*. Unlike conventional supervised models that directly predict explicit labels, reward models learn to assign scores that reflect the relative quality of different outputs. As a result, existing evaluation approaches mainly focus on measuring the preference modeling ability of reward models or their effectiveness in downstream RL optimization. Figure 5 compares three mainstream evaluation paradigms, including RL-based evaluation, pairwise ranking evaluation, and listwise ranking evaluation. We summarize these commonly used evaluation approaches as follows.

- **RL-based Evaluation.** A straightforward approach to evaluate a reward model is to measure its effectiveness in downstream RL training. Specifically, given multiple reward models $\{\mathcal{M}_r^1, \dots, \mathcal{M}_r^k\}$, we use each reward model to provide reward signals for training a policy model while keeping the training data, RL algorithm, and hyperparameters identical (Wang et al., 2026a; Frick et al., 2025). After RL training, we obtain a set of optimized policies $\{\mathcal{P}_1, \dots, \mathcal{P}_k\}$ and evaluate them on human preference benchmarks or downstream tasks. The performance of each policy then serves as an indirect measure of the quality of its corresponding reward model. In this way, we consider that a high-quality reward model should provide more reliable reward signals, enabling the policy to achieve better final performance.
- **Pairwise Ranking Evaluation.** Although RL-based evaluation directly measures whether a reward model can improve downstream policy optimization, it suffers from two limitations. First, it introduces significant evaluation costs. Since RL training requires substantial computational

Use each reward model to provide reward signals for RL training, and evaluate the trained policy on a benchmark. A higher benchmark score indicates that the corresponding reward model is better aligned with human preferences.

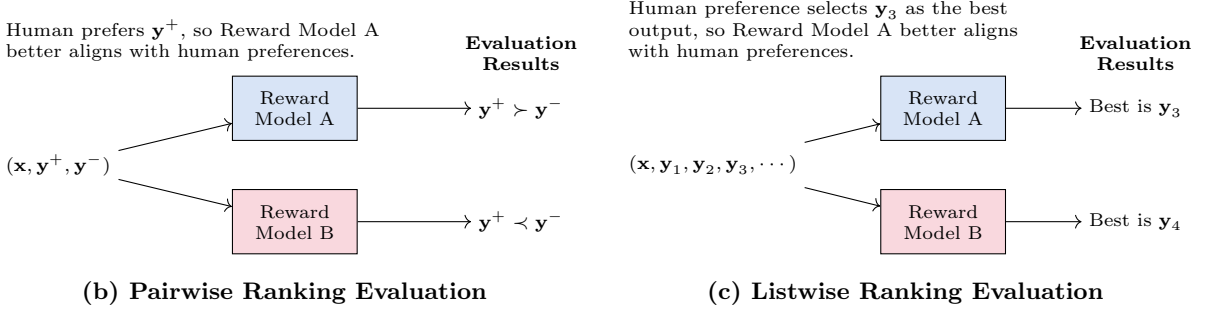
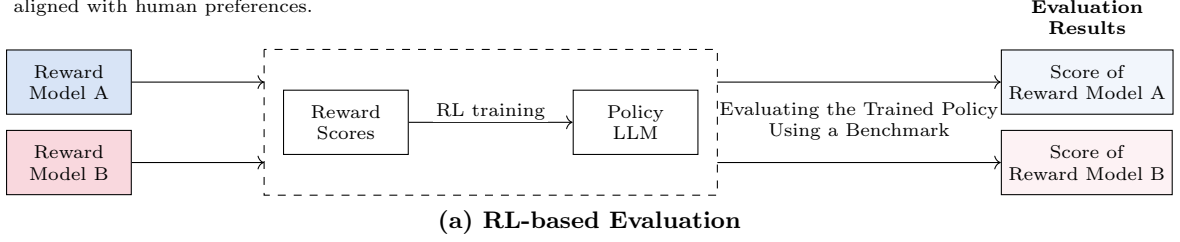


Figure 5: Commonly used reward model evaluation approaches. For the RL-based evaluation approach, the reward model is used to provide reward signals for policy optimization, and its quality is measured by the downstream performance of the optimized policy. For the pairwise evaluation approach, the reward model compares a preferred output with a dispreferred output, and its prediction is checked against human preference annotations. For the listwise evaluation approach, the reward model selects the best output from multiple candidates, and the selected output is compared with the human-preferred one.

resources and time, it is difficult to efficiently compare different reward models. Second, the evaluation results can be sensitive to other factors in the RL pipeline, such as the choice of RL algorithms, which may introduce additional variations beyond the quality of the reward model itself. To address these limitations, a more efficient approach is to directly evaluate the preference ranking ability of reward models. Given a prompt $\mathbf{x}$ and two candidate outputs $\mathbf{y}^+$ and $\mathbf{y}^-$, where $\mathbf{y}^+$ is preferred over $\mathbf{y}^-$ according to human preference annotations, the reward model predicts their relative preference. Specifically, for discriminative reward models, we compare the predicted scores of the two outputs. If the reward model assigns a higher score to $\mathbf{y}^+$, its prediction is consistent with human preference. For generative reward models, the model directly selects $\mathbf{y}^+$ as the preferred output based on its generated preference. Otherwise, the prediction is considered incorrect. By constructing a large number of pairwise ranking samples, we can evaluate the reward model using ranking accuracy:

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^N \mathbb{I} [R_{\theta}(\mathbf{x}_i, \mathbf{y}_i^+) > R_{\theta}(\mathbf{x}_i, \mathbf{y}_i^-)] \quad (10)$$

where N denotes the number of evaluation samples. Examples of this evaluation approach include RewardBench (Lambert et al., 2025; Malik et al., 2026), RM-Bench (Liu et al., 2025), RMB (Zhou et al., 2025), and JudgeBench (Tan et al., 2025).

- **Listwise Ranking Evaluation.** In practical alignment scenarios, the reward model often needs to select the best output from multiple candidates, such as in best-of- n sampling or reranking. Therefore, listwise ranking evaluation measures whether a reward model can produce a consistent ranking over a set of candidate outputs. Specifically, given a prompt $\mathbf{x}$ and a candidate output set: $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n\}$, the reward model assigns scores to all candidates. Based on the computed

rewards, the reward model selects the highest-scoring output:

$$\mathbf{y}_{\text{best}} = \arg \max_{\mathbf{y}_i \in \mathcal{Y}} R_{\theta}(\mathbf{x}, \mathbf{y}_i) \quad (11)$$

The selected output is then compared with the human-preferred output to evaluate how well the reward model captures human preferences. We take PPE (Frick et al., 2025) as an example to illustrate this evaluation approach. Specifically, we can construct each evaluation sample with multiple candidate outputs, where the quality of each output can be automatically verified. For example, in mathematical reasoning tasks, we can use an answer verifier to determine whether each generated solution is correct. Given a set of candidates $\mathcal{Y}$, the verification result of each output can be represented as:

$$v_i = \text{Verify}(\mathbf{x}, \mathbf{y}_i), \quad v_i \in \{0, 1\} \quad (12)$$

where $v_i = 1$ indicates that the output is correct. If the output selected by the reward model is verified as correct: $\text{Verify}(\mathbf{x}, \mathbf{y}_{\text{best}}) = 1$, the reward model is considered successful on this instance. The final performance is measured by the selection accuracy over all evaluation instances:

$$\text{Acc} = \frac{1}{N} \sum_{j=1}^N \mathbb{I}(\text{Verify}(\mathbf{x}_j, \mathbf{y}_{\text{best}}^j) = 1) \quad (13)$$

where N denotes the number of evaluation instances.

4.2 Better Advantage Estimation

Accurate advantage estimation is critical in RL, particularly when optimizing the policy model to truly reflect the potential benefits of different actions (or tokens in training LLMs). In this subsection, we will delve into methods for improving advantage estimation, providing more accurate advantages in the process of optimizing the policy model.

4.2.1 Temporal Difference-based Advantage Estimation

Let us first review the Monte Carlo-based advantage estimation. As discussed in Section ??, outputs are sampled and their values computed. The advantage can be obtained by comparing the actual received rewards with the predicted values at time step t :

$$A_t = \sum_{k=t}^T r_k - V_t \quad (14)$$

While the Monte Carlo-based estimation of advantage provides a relatively stable training process, there are two main challenges associated with it:

- Sampling an entire output for each input is necessary. In some cases where immediate rewards are available, it could be more efficient to update the policy model with partial output. Unfortunately, when using Eq. (14) for advantage estimation, this becomes unfeasible. This is because we must compute the term $\sum_{k=t}^T r_k$, which requires the rewards of the entire output.
- This method still results in high variance. Since a single sample operation might be influenced by random factors, the estimated results may not be stable, as illustrated in Figure 6.

To address these challenges, an alternative approach is the temporal difference-based advantage estimation (Sutton, 1988), which allows for estimating the advantage using incomplete outputs. More specifically, this method computes the difference between the predicted values at consecutive time steps (i.e., current and next steps), adjusting the advantage estimate based on new information as it becomes available, thus

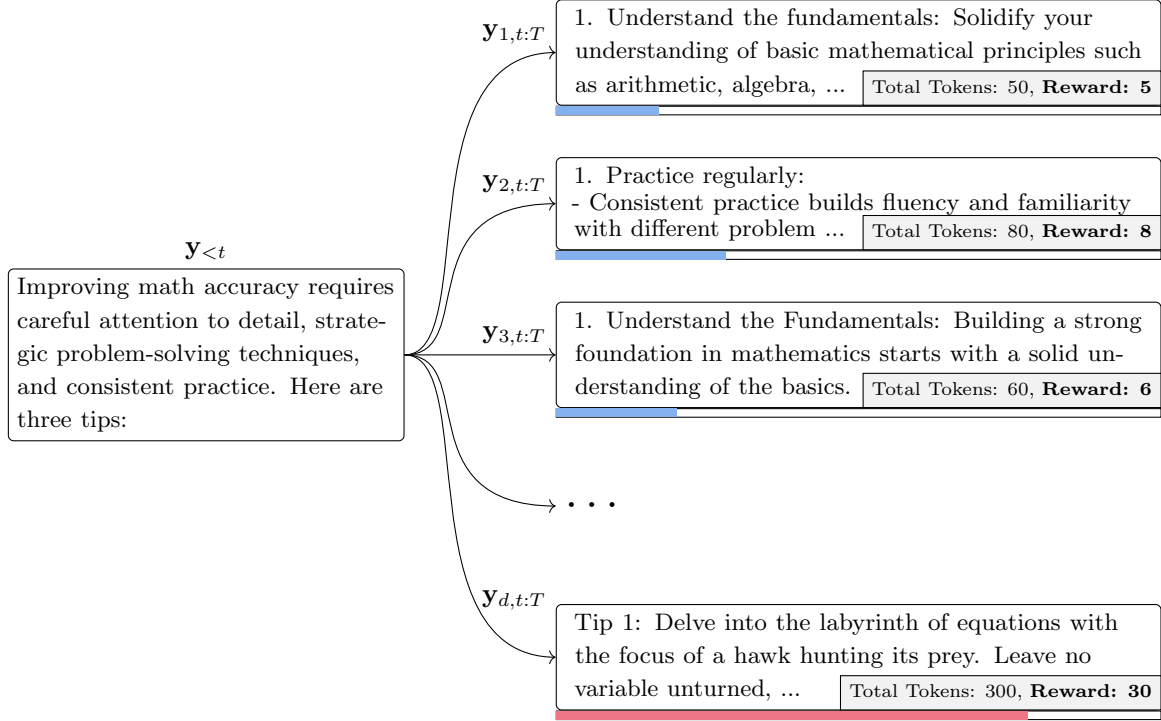


Figure 6: Illustration of the issue of high variance in Monte Carlo-based advantage estimation, using a length-based reward function as described in Eq. (??). Typically, if the value model is well-trained, it would predict an average V_t of 6 at time step t , based on most outputs having around 60 tokens in length. However, for outlier outputs like $\mathbf{y}_{d,t:T}$, which might extend up to 300 tokens, the advantage estimation can exhibit high variance. For instance, the advantage for such an outlier could be computed as $A_t(\mathbf{x}, \mathbf{y}_{d,<t}, y_{d,t}) = \sum_{k=t}^T r_k - V_t = 30 - 6 = 24$, a stark contrast to another output, say $\mathbf{y}_{1,t:T}$, which is closer to the average length, where the advantage might be $A_t(\mathbf{x}, \mathbf{y}_{1,<t}, y_{1,t}) = -1$. This discrepancy leads to high gradient variance, potentially destabilizing the learning process.

reducing the dependency on the entire output. In a practical implementation, the temporal difference-based advantage estimation can be given by

$$A_t^{\text{TD}} = r_t + \gamma V_{t+1} - V_t \quad (15)$$

By focusing on the current and next steps, temporal difference-based advantage estimation minimizes the influence of distant future events. This leads to lower variance in optimizing the policy model and improves the stability of policy training.

4.2.2 Generalized Advantage Estimation

While temporal difference-based estimation effectively reduces variance, it can also increase estimation bias due to its heavy reliance on the predictions of the value model. By contrast, one effective method is generalized advantage estimation (GAE), which is one of the most popular advantage estimation methods used in LLMs and other fields (Schulman et al., 2016). This method mainly refines the advantage estimation by using exponentially weighted averages of temporal-difference residuals across multiple future steps. More specifically, consider the temporal-difference residual δ_t at time step t , defined as

$$\delta_t = r_t + \gamma V_{t+1} - V_t \quad (16)$$

GAE introduces parameters that dynamically balance the trade-off between bias and variance. This basic idea is to use the weighted sum of multi-step temporal-difference residuals as an estimation of the advantage,

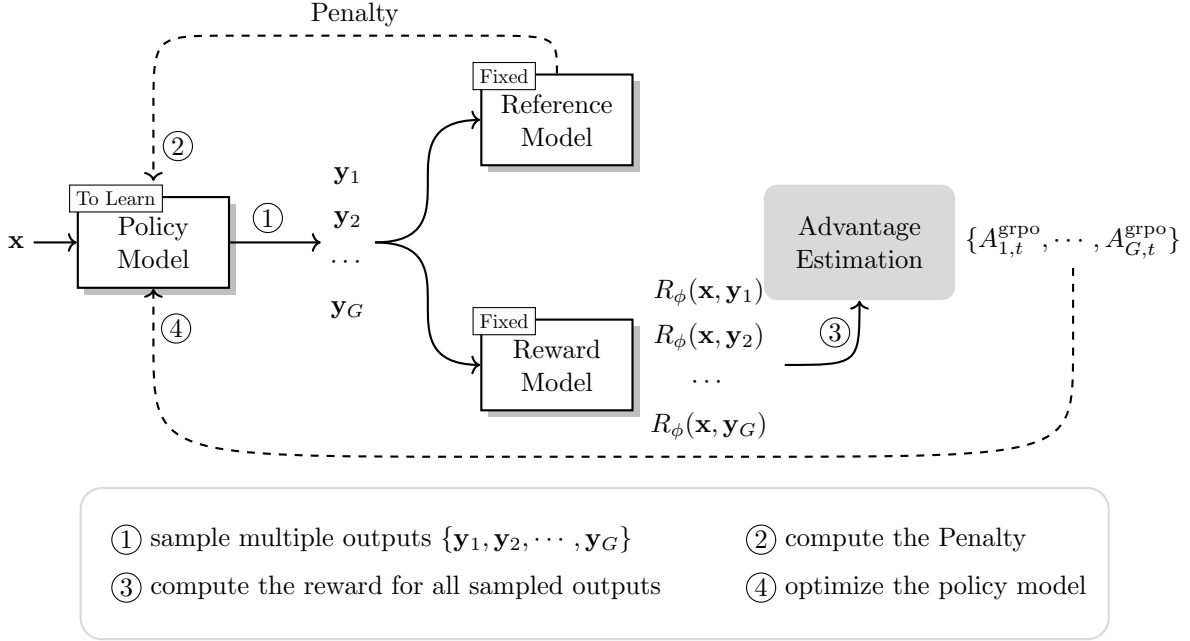


Figure 7: Group relative policy optimization. Compared to PPO, it foregoes the value model by estimating advantages directly from the rewards of a group of outputs. This method can simplify the training process and reduce the computational resources.

combining the advantages of Monte Carlo (high variance and low bias) and temporal difference methods (low variance and high bias). This combination can be expressed as

$$A_t^{\text{gae}} = \sum_{n=0}^{T-t} (\gamma\lambda)^n \delta_{t+n} \quad (17)$$

We can recursively develop the above model:

$$A_t^{\text{gae}} = \delta_t + \gamma\lambda A_{t+1}^{\text{gae}} \quad (18)$$

This recursive formulation shows that the current advantage estimate incorporates not only the immediate temporal difference error at time t but also the estimated advantages of future time steps, weighted and adjusted by the parameters γ and λ . This method effectively combines the strengths of both Monte Carlo and temporal difference methods, leading to a more stable and efficient training process.

4.2.3 Group Relative Policy Optimization

Although the methods discussed in the previous subsections effectively estimate advantage during the learning process, they all rely on a value model that is typically trained concurrently with the policy model. This dependence not only complicates the training process but also requires significant computational resources when applied to training LLMs. Given this, a natural idea arises: could we use more lightweight methods to compute the advantage without relying on the value model? By doing so, we could forego the value model component, thereby improving efficiency.

In fact, this is entirely feasible, and one example of such an approach is [Shao et al. \(2024\)](#)’s approach, called group relative policy optimization or GRPO for short. In the GRPO approach, the advantage is computed based on the relative rewards of the outputs within each group. More specifically, given an input $\mathbf{x}$, GRPO samples a group of outputs $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_G\}$. Then, the rewards for each output are computed using a

reward model or a reward function, denoted as $\mathbf{R} = \{R(\mathbf{x}, \mathbf{y}_1), R(\mathbf{x}, \mathbf{y}_2), \dots, R(\mathbf{x}, \mathbf{y}_G)\}$. The advantage of the i -th output can be computed through a relative reward in comparison to the other outputs:

$$A_{i,t}^{\text{grpo}} = \frac{R(\mathbf{x}, \mathbf{y}_i) - \text{Mean}(\mathbf{R})}{\text{Std}(\mathbf{R})} \quad (19)$$

where $\text{Mean}(\cdot)$ and $\text{Std}(\cdot)$ represent the mean and standard deviation functions, respectively. Note that the advantage computation used here differs slightly from that in PPO. In this case, this computed advantage is applied to each time step, whereas in the traditional PPO, we separately compute an advantage for each time step. At this point, we can also use the process reward model to supplement the advantage computation, allowing us to compute an advantage specific to each time step. A simple way to achieve this is by implementing Eq. (19) at each time step, where the rewards are computed using the process reward model. As a result, the objective for GRPO can be defined according to Eq. (??):

$$\mathcal{L}_g(\theta) = -\mathbb{E}_{\mathbf{x} \sim \mathcal{S}_x} \frac{1}{G} \sum_{i=1}^G \left[\sum_{t=1}^T \text{Clip} \left(\frac{\Pr_{\theta}(y_{i,t} | \mathbf{x}, \mathbf{y}_{i,<t})}{\Pr_{\theta_{\text{ref}}}(y_{i,t} | \mathbf{x}, \mathbf{y}_{i,<t})} A_{i,t}^{\text{grpo}} \right) - \beta \text{Penalty} \right] \quad (20)$$

In addition to optimizing the advantage computation, GRPO modifies the penalty term to enhance the optimization objective further. However, since our focus here is on advantage estimation, we will not delve into its details. Interested readers can refer to the GRPO paper for further details. The workflow of GRPO is illustrated in Figure 7, showcasing how these modifications integrate into the training process.

Moreover, there are other methods like GRPO that design advantage estimation without relying on a value model for training LLMs, such as those discussed in Li et al. (2024) and Hu (2025).

4.3 Efficient RL Methods

Efficiency is a critical consideration for many practical applications of RL, and it becomes even more significant in LLM training due to their vast number of parameters. For example, we might wish to train LLMs using RL, given memory and time constraints. In practice, the efficiency of RL is not a single issue but encompasses a wide range of challenges. While these challenges can be categorized in various ways in the existing literature, we focus on two fundamental aspects: *time* and *space* efficiency, which are commonly considered in efficiency-related problems. For these two efficiency problems, we aim to achieve the desired RL performance with the least amount of time and memory required.

In this section, we will not discuss all the issues related to the efficiency of RL, which is an extensive topic. Instead, we will focus on the commonly used efficient methods for training LLMs with RL. Some of these methods refine the sampling process, while others aim to eliminate certain components, such as the reward model, and utilize alternative lightweight methods in their place. Nonetheless, although these efficient methods are suggested for training LLMs, they are rather general and can be utilized in other RL scenarios.

4.3.1 Dynamic Sampling

As mentioned in Section ??, training LLMs with RL typically requires sampling for each sample $\mathbf{x}$ in $\mathcal{S}_x$, which introduces significant computational time overhead. This becomes particularly challenging in large-scale RL scenarios, where thousands of training steps need to be conducted, such as in DeepSeek-R1 (Guo et al., 2025).

From the perspective of LLM inference, there are many methods to reduce the time overhead of sampling: since the sampling process is implemented by LLM inference, we have reason to believe that any method that accelerates inference can also be applied to reduce the time required for sampling. Examples include KV-Cache (Pope et al., 2023), quantization (Zhao et al., 2024), and speculative decoding (Chen et al., 2023). However, in this section, we will not discuss these methods in detail, as there are numerous approaches, each addressing different aspects of LLM inference. We refer the interested readers to these papers for more

details. Instead, we will focus on one particular method that aims to reduce sampling overhead from the perspective of optimizing RL for training LLMs.

Before discussing specific methods, let us first review the purpose of sampling. Given a sample $\mathbf{x}$, the goal of sampling is to explore a better output $\mathbf{y}$ from the policy model. This process allows us to evaluate different possible outputs in order to find those that lead to improved outcomes, enabling the model to make more informed decisions and optimize its performance on the state. Unlike typical RL scenarios, where policy models may start with limited information or random initialization, the policy model in LLM training is usually quite advanced. These LLMs often begin as pre-trained models equipped with substantial knowledge acquired through pre-training and SFT. As a result, for certain samples, we can consider that the policy model may already perform well without the need for further exploration and optimization. This can be viewed through the lens of the classic RL dilemma of exploration versus exploitation (Sutton & Barto, 2018). In such cases, exploration may be less necessary, and exploiting the knowledge already embedded in the pre-trained model could be more effective. Thus, a balance must be struck between exploring new possibilities and leveraging the pre-existing knowledge of the policy model to maximize efficiency.

To achieve this balance between exploration and exploitation, one simple and straightforward approach to improving the efficiency of sampling is to identify the samples that are required to explore and perform sampling only for those samples. Given an input-only dataset $\mathcal{S}_x = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where N denotes the size of the dataset, we can use a reward model to evaluate the need for exploration by the policy model. First, we generate $\hat{\mathbf{y}}$ for each $\mathbf{x}$ using greedy search. Then, by taking $\mathbf{x}$ and $\hat{\mathbf{y}}$ as input to the reward model, the reward for the k -th sample $\mathbf{x}_k$ is computed as $R_\phi(\mathbf{x}_k, \hat{\mathbf{y}}_k)$.

At this point, we can directly use the reward value to determine whether the policy model requires further exploration and optimization for the sample. Specifically, we set a threshold r_{upper} . If the reward for a sample exceeds r_{upper} , we can consider that no further sampling and optimization is required. Otherwise, exploration and optimization are required. This approach enables selective sampling and optimization, improving the efficiency of the overall process by focusing resources on the most promising samples that are likely to benefit from further exploration (Wang et al., 2024a). A challenge in implementation is determining the value of r_{upper} . Typically, we set it as the average reward:

$$r_{\text{upper}} = \frac{1}{N} \sum_{i=1}^N R_\phi(\mathbf{x}_i, \hat{\mathbf{y}}_i) \quad (21)$$

A similar dynamic sampling strategy, known as prioritized sampling, has proven effective in large-scale RL scenarios (Team et al., 2025). However, we must consider another important factor in dynamic sampling. For samples with very low rewards, the policy model may lack the capacity to learn and optimize effectively. Excessive exploration of such samples can be inefficient. Consequently, we can introduce an additional threshold r_{lower} to eliminate unnecessary exploration of these low-reward samples, thereby enhancing efficiency. In this case, the policy model can concentrate its exploration and optimization efforts on a select group of samples, avoiding wasted sampling time on those that are unlikely to explore a more optimal output (Li et al., 2025).

A concept closely related to the discussion here is sample efficiency. This topic has been widely discussed in recent literature. For example, some research on knowledge distillation seeks to select a smaller subset of samples to achieve more optimal performance (Wang et al., 2021). More recently, much work on instruction sample selection in SFT aims to improve SFT efficiency by learning from a small number of instruction samples (Chen et al., 2024).

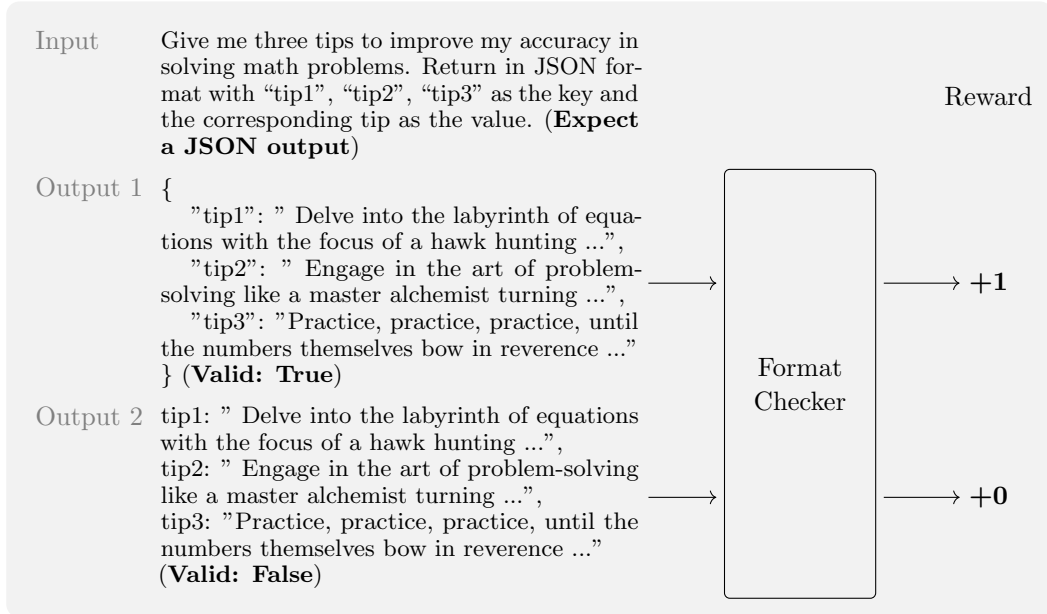
4.3.2 Lightweight Reward Methods

An additional computational overhead in training LLMs with RL is the frequent need to call the reward model to compute rewards. Furthermore, to ensure the reliability and generalizability of the reward model,

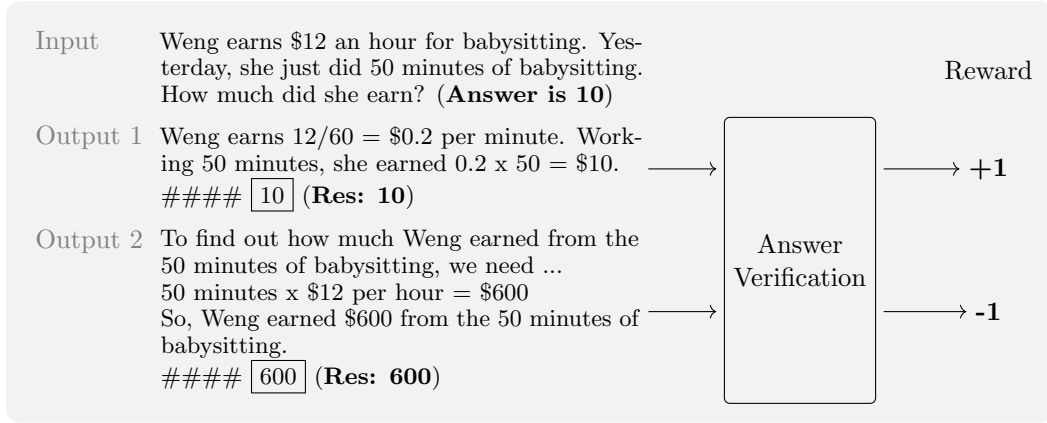
we often scale it to a large size (Gao et al., 2023), which increases the computational time for reward computations.

One approach for mitigating this issue is to explore rule-based rewards, for example, as discussed in Section ??, the length of the outputs could be utilized as a reward. Instead of relying on large-scale, resource-intensive models, rule-based rewards assign rewards using predefined and task-specific rules. These rules, often derived from expert knowledge or task-specific requirements, are computationally inexpensive and provide fast, effective feedback to the policy model. By replacing or complementing traditional reward models with rule-based approaches, we can reduce both the time and computational cost associated with reward computations while still offering meaningful guidance for the learning process. This approach is also based on the understanding that, in some tasks, human preferences are simple to describe and do not require the complexity of training a reward model. Instead, we can use rules to express these preferences efficiently.

Here, in addition to the previously mentioned length-based rewards, we further demonstrate two examples of rule-based rewards. First, considering the output format, we can design rewards that encourage the policy model to generate outputs adhering to specific formats. For example, in response to the input “Give me three tips to improve my accuracy in solving math problems,” we can assign rewards based on whether the output can be parsed as JSON correctly and include “tip1”, “tip2”, and “tip3” as the keys.



Similarly, in mathematical reasoning tasks, where the desired outcome is a correct final answer, we can design a rule to verify the correctness of the answer and provide rewards based on whether the result is correct (Havrilla et al., 2024; Shao et al., 2024).



In addition to improving computational efficiency, another benefit of using rule-based rewards is their stability. Compared to reward models, rule-based rewards are less prone to issues such as reward hacking. As the rules are predefined and not subject to the same complex training process as traditional models, the risk of misalignment between the optimization objectives of the policy model and the oracle rewards is reduced. This stability makes rule-based rewards a more predictable and reliable approach, ensuring that the learning process of the policy model is less influenced by prediction errors or biases that may arise from the reward model. Furthermore, recent research has shown that employing rule-based rewards can significantly enhance the reasoning capabilities of LLMs through RL, underscoring its practical effectiveness (Guo et al., 2025; Xie et al., 2025).

Another lightweight reward method is to use a reward model with fewer parameters, which would reduce both computational time and resource usage. There are several methods to achieve this. For example, given that reward models fundamentally utilize an LLM, many efficient methods, such as Mixture of Experts (MoE) (Masoudnia & Ebrahimpour, 2014) and model pruning (Sun et al., 2024), originally developed for LLMs can be adapted to the reward model to improve its efficiency. Furthermore, knowledge distillation techniques offer a pathway to construct a smaller reward model that learns to emulate the behavior of a larger one (Wang et al., 2023a). These approaches not only improve the efficiency of the reward model but also maintain its ability to deliver accurate rewards.

4.4 Direct Preference Optimization

Although learning reward models is a standard step in RL, it makes the entire training process much more complex than supervised training. Training a reliable reward model is itself not an easy task and a poorly trained reward model can greatly affect the outcome of policy learning. We now consider an alternative method, called direct preference optimization (DPO), which simplifies the training framework by eliminating the need to explicitly model rewards (Rafailov et al., 2023). This method directly optimizes the policy model based on user preferences rather than developing a separate reward model. As a result, we can achieve human preference alignment in a supervised learning-like fashion. Figure 8 shows a comparison of the standard PPO method and the DPO method.

DPO simplifies the RL process for LLMs by utilizing a straightforward cross-entropy loss, which streamlines the learning process and enhances its stability. Rather than delving into the detailed derivation of the DPO loss function, we will discuss its optimization objective from the perspective of optimizing an LLM. The loss function derived from a Bradley-Terry reward model can be given by

$$\mathcal{L}_{\text{dpo}}(\theta) = -\mathbb{E}_{(\mathbf{x}, \mathbf{y}_a, \mathbf{y}_b) \sim \mathcal{D}_r} \left[\log \text{Sigmoid} \left(\beta \log \frac{\Pr_{\theta}(\mathbf{y}_a | \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a | \mathbf{x})} - \beta \log \frac{\Pr_{\theta}(\mathbf{y}_b | \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b | \mathbf{x})} \right) \right] \quad (22)$$

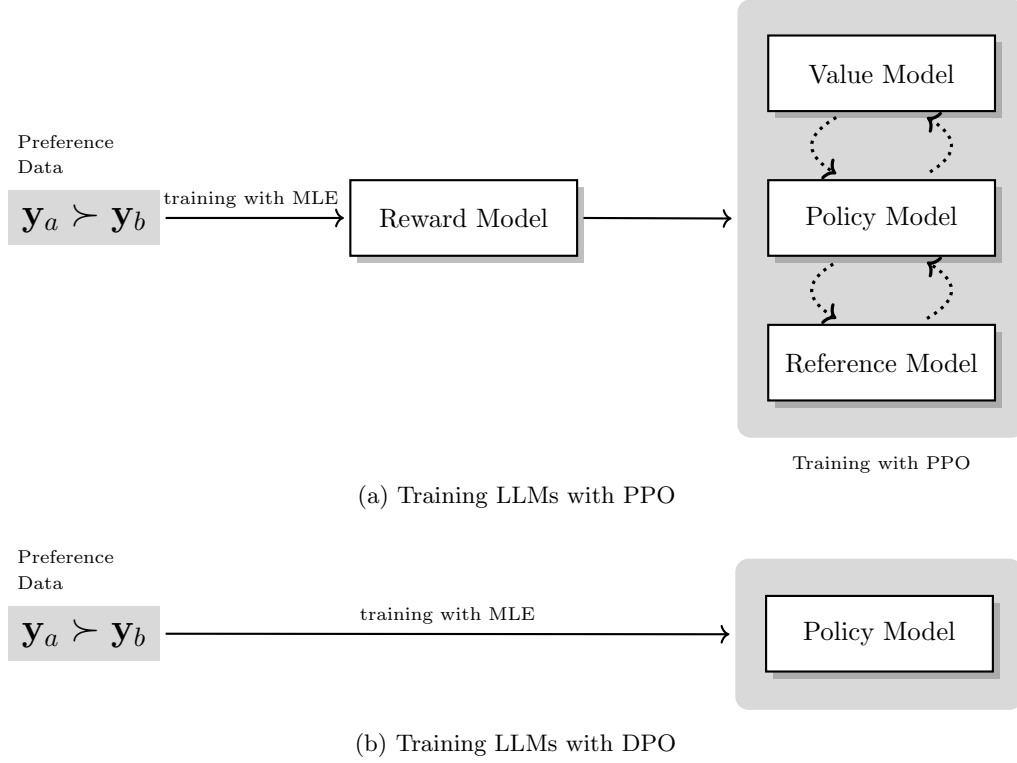


Figure 8: Standard PPO vs. DPO in training LLMs. In PPO, the human preference data is used to train a reward model, which is then employed to train the policy and the value function. In DPO, the use of human preference data is more direct, and the policy is trained on this data without the need for reward model training.

This loss function utilizes the implicit reward for DPO training, which replaces the need for an external reward model. The implicit reward is computed as the log-ratio of probabilities:

$$R(\mathbf{x}, \mathbf{y}) = \beta \log \frac{\Pr_{\theta}(\mathbf{y}|\mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}|\mathbf{x})} \quad (23)$$

Here, we can consider that DPO directly models human feedback through the relative likelihood of outputs under the policy and reference models. This method provides a more straightforward approach than traditional methods that use a separate reward model and then apply RL algorithms like PPO to adjust the probabilities of sampled outputs. In this way, we can further understand the optimization objective of DPO by leveraging the hypothesis space as mentioned in Section ?? . In practice, as illustrated in Figure 9, we can see DPO as reranking in the hypothesis space through the Bradley-Terry model approach. Specifically, this method increases the probabilities of preferred outputs in preference data while decreasing those of dispreferred ones. Similar to TRPO, DPO also incorporates a penalty derived from the reference model, which ensures that updates remain within a trusted behavior space for the policy model.

However, there are two sides to every coin. While DPO simplifies the RL process, it also introduces limitations. Notably, since DPO eliminates the exploration phase during training, its potential peak performance is generally considered lower compared to RL-based methods like PPO, which incorporate exploration to discover more effective policies. In other words, the performance of DPO is inherently bounded by the labeled preferred outputs in the preference data. To address this limitation, current approaches focus on ensuring high-quality preferred outputs, either by employing advanced LLMs like GPT-4 or through human labeling (Cui et al., 2024a; Morimura et al., 2024).

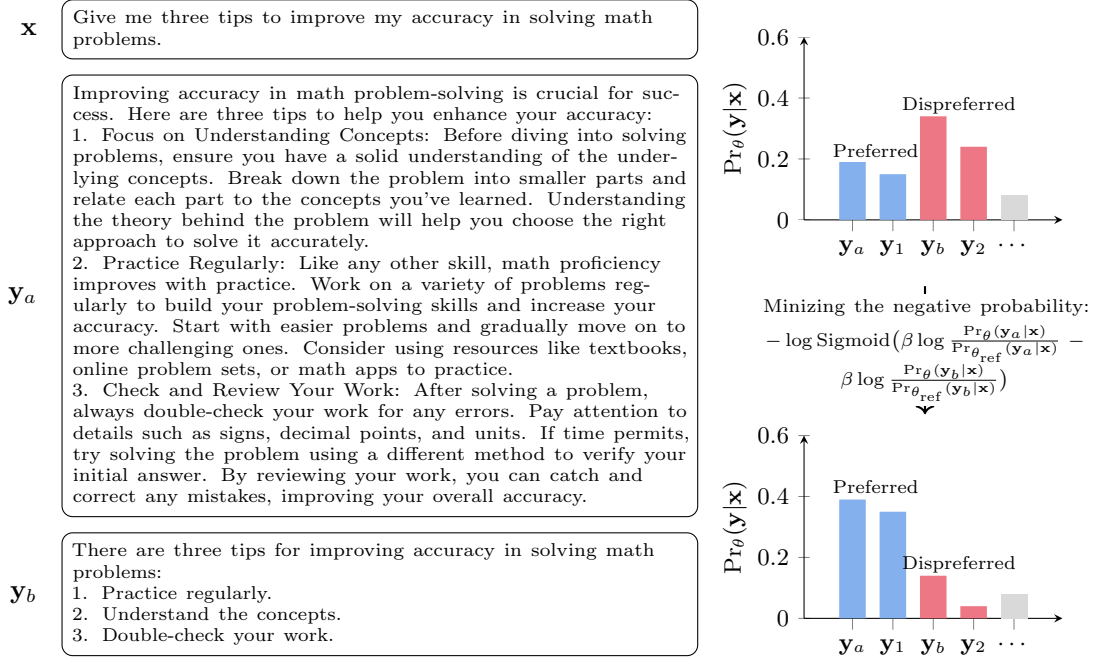


Figure 9: Illustration of the optimization objective of DPO from a hypothesis space perspective. The right-top chart shows the probability distributions over different hypotheses before optimization, where $\mathbf{y}_a$ is the preferred output and $\mathbf{y}_b$ is the dispreferred output. The right-bottom chart shows the probabilities after applying DPO, where the likelihood of the preferred output increases and that of dispreferred outputs decreases. Note that during this optimization process, outputs similar to the preferred output $\mathbf{y}_a$ (such as $\mathbf{y}_1$) will obtain an increased probability, while those similar to the less preferred output $\mathbf{y}_b$ (such as $\mathbf{y}_2$) will experience a decreased probability, due to the generalization of the model.

Another limitation associated with DPO is over-optimization. Since DPO employs the Bradley-Terry model for modeling preferences, it can suffer from over-optimization, such as length exploitation, where a longer output might be mistakenly deemed as more aligned with human preferences (Singhal et al., 2023; Wang et al., 2023b). Many efforts have been made to address this issue and propose different variants of DPO, as detailed in Table 1. Note that here we only provide an introduction to their optimization objectives. For more discussions on these variants, interested readers can refer to the related papers.

Method	Objective
IPO (Azar et al., 2024)	$\left(\log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})} - \frac{1}{2\tau}\right)^2$
CPO (Xu et al., 2024)	$-\log \text{Sigmoid}(\beta \log \Pr_\theta(\mathbf{y}_a \mathbf{x}) - \beta \log \Pr_\theta(\mathbf{y}_b \mathbf{x})) - \lambda \log \Pr_\theta(\mathbf{y}_a \mathbf{x})$
KTO (Ethayarajh et al., 2024)	$-\lambda_a \text{Sigmoid}\left(\beta \log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - z_{\theta_{\text{ref}}}\right) + \lambda_b \text{Sigmoid}\left(z_{\theta_{\text{ref}}} - \beta \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})}\right)$ where $z_{\theta_{\text{ref}}} = \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathcal{S}} [\beta \text{KL}(\Pr_\theta(\mathbf{y} \mathbf{x}) \Pr_{\theta_{\text{ref}}}(\mathbf{y} \mathbf{x}))]$
ORPO (Hong et al., 2024)	$-\log p_\theta(\mathbf{y}_a \mathbf{x}) - \lambda \log \text{Sigmoid}\left(\log \frac{p_\theta(\mathbf{y}_a \mathbf{x})}{1-p_\theta(\mathbf{y}_a \mathbf{x})} - \log \frac{p_\theta(\mathbf{y}_b \mathbf{x})}{1-p_\theta(\mathbf{y}_b \mathbf{x})}\right)$ where $p_\theta(\mathbf{y} \mathbf{x}) = e^{\frac{1}{ \mathcal{Y} } \log \Pr_\theta(\mathbf{y} \mathbf{x})}$
R-DPO (Gallego, 2024)	$-\log \text{Sigmoid}\left(\beta \log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - \beta \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})} + (\alpha \mathbf{y}_a - \alpha \mathbf{y}_b)\right)$
PCDPO (Zhou et al., 2024)	$-\log \text{Sigmoid}(\Delta^* - \Delta_{\Pr_\theta}) - \log \text{Sigmoid} \Delta_{\Pr_\theta}$, where $\Delta_{\Pr_\theta} = \beta \log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - \beta \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})}$, $\Delta^* = \frac{\beta_1}{\text{Sim}(\mathbf{y}_a, \mathbf{y}_b \mathbf{x}) + \beta_2} + \beta_3$
SimPO (Meng et al., 2024)	$-\log \text{Sigmoid}\left(\frac{\beta}{ \mathbf{y}_a } \log \Pr_\theta(\mathbf{y}_a \mathbf{x}) - \frac{\beta}{ \mathbf{y}_b } \log \Pr_\theta(\mathbf{y}_b \mathbf{x}) - \gamma\right)$
ODPO (Amini et al., 2024)	$-\log \text{Sigmoid}\left(\beta \log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - \beta \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})} - \delta(\mathbf{x}, \mathbf{y}_a, \mathbf{y}_b)\right)$
Cal-DPO (Xiao et al., 2024)	$-\log \text{Sigmoid}(\Delta_{\Pr_\theta}) + \lambda \mathcal{L}_{\text{cal}}$ where $\Delta_{\Pr_\theta} = \beta \log \frac{\Pr_\theta(\mathbf{y}_a \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_a \mathbf{x})} - \beta \log \frac{\Pr_\theta(\mathbf{y}_b \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_b \mathbf{x})}$, $\mathcal{L}_{\text{cal}}$ calibrates implicit rewards.
TDPO (Zeng et al., 2024)	$-\log \text{Sigmoid}\left(\sum_{t=1}^{T_a} \beta \log \frac{\Pr_\theta(\mathbf{y}_{a,t} \mathbf{x}, \mathbf{y}_{a,<t})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_{a,t} \mathbf{x}, \mathbf{y}_{a,<t})} - \sum_{t=1}^{T_b} \beta \log \frac{\Pr_\theta(\mathbf{y}_{b,t} \mathbf{x}, \mathbf{y}_{b,<t})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_{b,t} \mathbf{x}, \mathbf{y}_{b,<t})}\right) + \lambda \sum_t \text{KL}_t$ where KL_t denotes the token-level KL regularization term.
AOT (Melnyk et al., 2024)	$\mathcal{L}_{\text{OT}}(\{r_\theta(\mathbf{x}, \mathbf{y}_a)\}, \{r_\theta(\mathbf{x}, \mathbf{y}_b)\})$ where $r_\theta(\mathbf{x}, \mathbf{y}) = \beta \log \frac{\Pr_\theta(\mathbf{y} \mathbf{x})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y} \mathbf{x})}$, and $\mathcal{L}_{\text{OT}}$ aligns preference distributions via optimal transport.
D2PO (Shao et al., 2025)	$-\log \text{Sigmoid}\left(\sum_{t=0}^T \gamma^t \beta \log \frac{\Pr_\theta(\mathbf{y}_{a,t} \mathbf{x}, \mathbf{y}_{a,<t})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_{a,t} \mathbf{x}, \mathbf{y}_{a,<t})} - \sum_{t=0}^T \gamma^t \beta \log \frac{\Pr_\theta(\mathbf{y}_{b,t} \mathbf{x}, \mathbf{y}_{b,<t})}{\Pr_{\theta_{\text{ref}}}(\mathbf{y}_{b,t} \mathbf{x}, \mathbf{y}_{b,<t})}\right)$

Table 1: DPO variants and their optimization objectives.

References

- Afra Amini, Tim Vieira, and Ryan Cotterell. Direct preference optimization with an offset. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9954–9972, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.findings-acl.592/>.
- Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Rémi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pp. 4447–4455. PMLR, 2024. URL <https://proceedings.mlr.press/v238/gheshlaghi-azar24a.html>.
- Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL <https://openreview.net/forum?id=SJDaqqveg>.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL <https://arxiv.org/abs/2302.01318>.
- Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpargasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=FdVXgSJhvvz>.
- Thomas Coste, Usman Anwar, Robert Kirk, and David Krueger. Reward model ensembles help mitigate overoptimization. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=dcjtMYkpXx>.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. ULTRA FEEDBACK: boosting language models with scaled AI feedback. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a. URL <https://openreview.net/forum?id=BOorDpKHjJ>.
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. ULTRA FEEDBACK: boosting language models with scaled AI feedback. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024b. URL <https://openreview.net/forum?id=BOorDpKHjJ>.
- Hongxin Ding, Baixiang Huang, Yue Fang, Weibin Liao, Zheng Li, Jinyang Zhang, Zhijing Wu, Junfeng Zhao, and Yasha Wang. Evorubrics: Dynamic rubrics as rewards via adversarial co-evolution for llm reinforcement learning, 2026. URL <https://arxiv.org/abs/2606.23038>.
- Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA*,

- USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/5fc47800ee5b30b8777fdd30abcaaf3b-Abstract-Conference.html.
- Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D’Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, et al. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking, 2023. URL <https://arxiv.org/abs/2312.09244>.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024. URL <https://arxiv.org/abs/2402.01306>.
- Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for RLHF. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=cbttLt094Q>.
- Víctor Gallego. Refined direct preference optimization with synthetic data for behavioral alignment of llms, 2024. URL <https://arxiv.org/abs/2402.08005>.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023. URL <https://proceedings.mlr.press/v202/gao23h.html>.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskiy, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning, 2024. URL <https://arxiv.org/abs/2403.04642>.
- Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, Miami, Florida, USA, 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.emnlp-main.626/>.
- Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models, 2025. URL <https://arxiv.org/abs/2501.03262>.
- Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal M. P. Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=CjwERcAU7w>.
- Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL <https://aclanthology.org/2025.findings-naacl.96/>.
- Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling, 2025. URL <https://arxiv.org/abs/2502.11886>.

- Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=Stn8hXkpe6>.
- Tianci Liu, Ran Xu, Tony Yu, Ilgee Hong, Carl Yang, Tuo Zhao, and Haoyu Wang. OpenRubrics: Towards scalable synthetic rubric generation for reward modeling and LLM alignment. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17417–17437, San Diego, California, United States, 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. URL <https://aclanthology.org/2026.acl-long.791/>.
- Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, 2023. Association for Computational Linguistics. URL <https://aclanthology.org/2023.emnlp-main.153/>.
- Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. Rm-bench: Benchmarking reward models of language models with subtlety and style. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=QEHrmQPBdd>.
- Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models, 2024. URL <https://arxiv.org/abs/2410.12832>.
- Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah Smith, Hanna Hajishirzi, and Nathan Lambert. Rewardbench 2: Advancing reward model evaluation. In *International Conference on Learning Representations*, volume 2026, pp. 144839–144866, 2026.
- Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.
- Igor Melnyk, Youssef Mroueh, Brian Belgodere, Mattia Rigotti, Apoorva Nitsure, Mikhail Yurochkin, Kristjan H. Greenewald, Jirí Navrátil, and Jarret Ross. Distributional preference alignment of llms via optimal transport. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/bce96750dcb64f3257ceabdf0b9c9bf-Abstract-Conference.html.
- Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/e099c1c9699814af0be873a175361713-Abstract-Conference.html.
- Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.
- Tetsuro Morimura, Mitsuki Sakamoto, Yuu Jinnai, Kenshi Abe, and Kaito Ariu. Filtered direct preference optimization. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22729–22770, Miami, Florida, USA, 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.emnlp-main.1266/>.

- Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In Dawn Song, Michael Carbin, and Tianqi Chen (eds.), *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d24cdfb45e3d06dbfca2780-Abstract-mlsys2023.html.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.
- MohammadHossein Rezaei, Robert Vacareanu, Zihao Wang, Clinton Wang, Yunzhong He, and Afra Feyza Akyürek. Online rubrics elicitation from pairwise comparisons, 2025. URL <https://arxiv.org/abs/2510.07284>.
- John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL <http://arxiv.org/abs/1506.02438>.
- Ruichen Shao, Bei Li, Gangao Liu, Yang Chen, ZhouXiang, Jingang Wang, Xunliang Cai, and Peng Li. Earlier tokens contribute more: Learning direct preference optimization from temporal decay perspective. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=OspqtLVUN5>.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- William F. Shen, Xinchu Qiu, Chenxi Whitehouse, Lisa Alazraki, Shashwat Goel, Francesco Barbieri, Timon Willi, Akhil Mathur, and Ilias Leontiadis. Rethinking rubric generation for improving llm judge and reward modeling for open-ended tasks, 2026. URL <https://arxiv.org/abs/2602.05125>.
- Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. A long way to go: Investigating length correlations in rlhf, 2023. URL <https://arxiv.org/abs/2310.03716>.
- Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=PxoFut3dWW>.
- Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (2nd ed.)*. The MIT Press, 2018.
- Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca A. Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=G0dksFayVq>.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms, 2025. URL <https://arxiv.org/abs/2501.12599>.

- Chenglong Wang, Hang Zhou, Kaiyan Chang, Tongran Liu, Chunliang Zhang, Quan Du, Tong Xiao, and Jingbo Zhu. Learning evaluation models from large language models for sequence generation, 2023a. URL <https://arxiv.org/abs/2308.04386>.
- Chenglong Wang, Hang Zhou, Yimin Hu, Yifu Huo, Bei Li, Tongran Liu, Tong Xiao, and Jingbo Zhu. ESRL: efficient sampling-based reinforcement learning for sequence generation. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19107–19115. AAAI Press, 2024a. URL <https://doi.org/10.1609/aaai.v38i17.29878>.
- Chenglong Wang, Yang Gan, Yifu Huo, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Tong Xiao, Chunliang Zhang, Tongran Liu, and Jingbo Zhu. GRAM: A generative foundation reward model for reward generalization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL <https://proceedings.mlr.press/v267/wang25ad.html>.
- Chenglong Wang, Yifu Huo, Yang Gan, Yongyu Mu, Qiaozhi He, Murun Yang, Bei Li, Chunliang Zhang, Tongran Liu, Anxiang Ma, Zhengtao Yu, Jingbo Zhu, and Tong Xiao. Probing preference representations: A multi-dimensional evaluation and analysis method for reward models. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 33404–33412. AAAI Press, 2026a. URL <https://doi.org/10.1609/aaai.v40i39.40627>.
- Chenglong Wang, Yongyu Mu, Hang Zhou, Yifu Huo, Ziming Zhu, Jiali Zeng, Murun Yang, Bei Li, Xiaoyang Hao, Chunliang Zhang, Fandong Meng, Jingbo Zhu, and Tong Xiao. Gram-r²: Self-training generative foundation reward models for reward reasoning. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 33395–33403. AAAI Press, 2026b. URL <https://doi.org/10.1609/aaai.v40i39.40626>.
- Chenglong Wang, Ziming Zhu, Yifu Huo, Bei Li, Qiaozhi He, Yan Ding, Xiaoyang Hao, Yuxin Gao, Tianhua Zhou, Xiaojia Chang, Tongran Liu, and Jingbo Zhu. Rrc: Unlocking generative reward models in llm reinforcement learning via ranking-based reward construction, 2026c. URL <https://arxiv.org/abs/2608.06310>.
- Fusheng Wang, Jianhao Yan, Fandong Meng, and Jie Zhou. Selective knowledge distillation for neural machine translation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6456–6466, Online, 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.acl-long.504/>.
- Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, 2024b. Association for Computational Linguistics. URL <https://aclanthology.org/2024.acl-long.511/>.
- Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. How far can camels go? exploring

- the state of instruction tuning on open resources. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b. URL http://papers.nips.cc/paper_files/paper/2023/hash/ec6413875e4ab08d7bc4d8e225263398-Abstract-Datasets_and_Benchmarks.html.
- Zequi Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. Fine-grained human feedback gives better rewards for language model training. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/b8c90b65739ae8417e61eadb521f63d5-Abstract-Conference.html.
- Teng Xiao, Yige Yuan, Huaisheng Zhu, Mingxiao Li, and Vasant G. Honavar. Cal-dpo: Calibrated direct preference optimization for language model alignment. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/cf8b2205e39f81726a8d828ecbe00ad0-Abstract-Conference.html.
- Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL <https://arxiv.org/abs/2502.14768>.
- Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: Pushing the boundaries of LLM performance in machine translation. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=51iwkioZpn>.
- Ran Xu, Tianci Liu, Zihan Dong, Tony Yu, Ilgee Hong, Carl Yang, Linjun Zhang, Tao Zhao, and Haoyu Wang. Alternating reinforcement learning for rubric-based reward modeling in non-verifiable llm post-training, 2026. URL <https://arxiv.org/abs/2602.01511>.
- Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. Regularizing hidden states enables learning generalizable reward model for llms. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/71f7154547c748c8041505521ca433ab-Abstract-Conference.html.
- Fangxu Yu, Tao Feng, Dehai Min, Zinan Lin, Weijia Xu, Michael Xu, Philip S. Yu, Ge Liu, and Tianyi Zhou. Reinforcement learning with evolving rubrics as rewards for audio reasoning, 2026. URL <https://arxiv.org/abs/2608.02831>.
- Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=1RZKuvqYCR>.
- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL <https://openreview.net/forum?id=Ccwp4tFtE>.

- Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL <https://openreview.net/forum?id=Ccwp4tFEtE>.
- Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In Phillip B. Gibbons, Gennady Pekhimenko, and Christopher De Sa (eds.), *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/5edb57c05c81d04beb716ef1d542fe9e-Abstract-Conference.html.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.
- Enyu Zhou, Guodong Zheng, Binghai Wang, Zhiheng Xi, Shihan Dou, Rong Bao, Wei Shen, Limao Xiong, Jessica Fan, Yurong Mou, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. RMB: comprehensively benchmarking reward models in LLM alignment. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=kmgrlG9TR0>.
- Hang Zhou, Chenglong Wang, Yimin Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. Prior constraints-based reward model training for aligning large language models. In Sun Maosong, Liang Jiye, Han Xianpei, Liu Zhiyuan, and He Yulan (eds.), *Proceedings of the 23rd Chinese National Conference on Computational Linguistics (Volume 1: Main Conference)*, pp. 1395–1407, Taiyuan, China, 2024. Chinese Information Processing Society of China. URL <https://aclanthology.org/2024.ccl-1.107/>.