# Improving LLM Generation with Inverse and Forward Alignment: Reward Modeling, Prompting, Fine-Tuning, and Inference-Time Optimization

**Hao Sun, Thomas Pouplin, Nicolás Astorga, Tennison Liu, Mihaela van der Schaar**
Department of Applied Mathematics and Theoretical Physics
University of Cambridge
Cambridge, UK
`{hs789, tp531, nja46, tl522, mv472}@cam.ac.uk`

## Abstract

Large Language Models (LLMs) are often characterized as samplers or generators in the literature, yet maximizing their capabilities in these roles is a complex challenge. Previous work has extensively explored the diverse applications of LLMs across various domains, including enhancing chat abilities, solving mathematical problems, adopting LLMs for evaluation, generating synthetic data, improving Bayesian optimization, and designing reward functions for reinforcement learning. Despite these advancements, key methods for improving LLM performance — such as prompt optimization, in-context learning, supervised fine-tuning, and reinforcement learning from human feedback—are typically studied in isolation.

In this work, we propose a unified optimization framework that encapsulates these diverse applications, providing a systematic approach to analyzing existing methods and uncovering potential improvements. We highlight (1) while LLMs **can** perform a wide range of tasks, truly **mastering** these tasks requires alignment, suggesting that any use of LLMs can benefit from alignment beyond mere sampling; (2) reward modeling is crucial for enhancing the effectiveness of LLMs, offering the **only viable path for inference-time optimization**; and (3) the choice of reward model depends on the specific **task properties and dataset availability**, necessitating careful consideration in its design.

## 1 Introduction: Alignment Improves LLMs from Capable to Matering

Large Language Models (LLMs) demonstrate a remarkable ***capability*** to perform a diverse class of tasks, underscoring their versatility and expanding utility across different domains. For instance, LLMs can serve as chat assistants, solve mathematical problems, design reward functions for control systems, enhance optimization techniques, and act as judges in evaluating various contexts. However, direct application of LLMs in those tasks always results in poor or sub-optimal performance, and eliciting their ***expertise*** requires non-trivial efforts.

Importantly, LLMs are recognized in the literature as potent samplers or generators within these areas. Addressing the challenge of maximizing their potential — transitioning their proficiency from merely ***Capable*** to ***Mastering*** various tasks — requires substantial effort. Despite extensive exploration of numerous applications, the lack of interconnectedness between different domains poses a significant challenge in translating successes from one task to another.

In this paper, we propose a unified optimization framework that brings together these diverse applications under a single perspective. By encapsulating these problems within a cohesive framework, we enable a systematic analysis of existing methods and identify opportunities for improvement. This

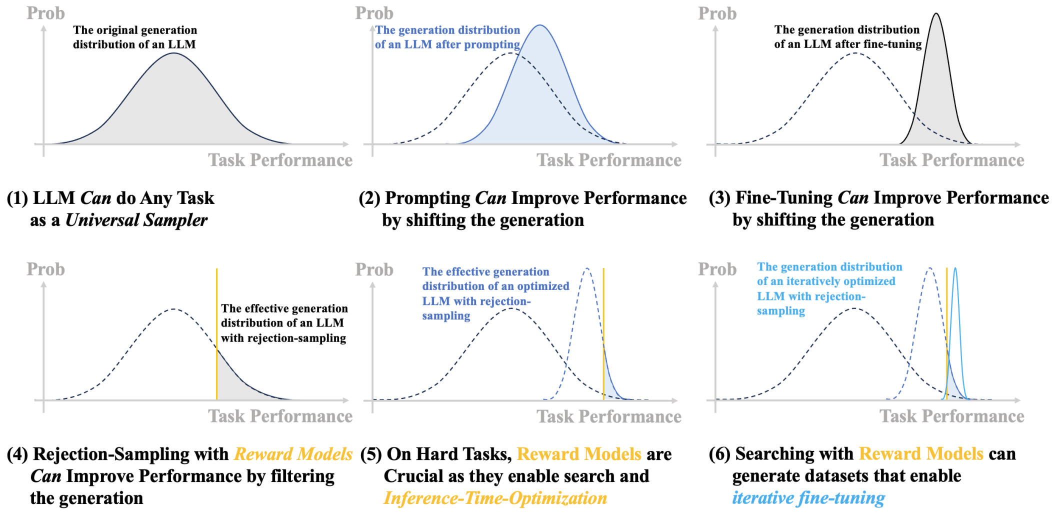
Prob
The original generation distribution of an LLM
Task Performance
(1) LLM *Can* do Any Task as a *Universal Sampler*
Prob
The generation distribution of an LLM after prompting
Task Performance
(2) Prompting *Can* Improve Performance by shifting the generation
Prob
The generation distribution of an LLM after fine-tuning
Task Performance
(3) Fine-Tuning *Can* Improve Performance by shifting the generation
Prob
The effective generation distribution of an LLM with rejection-sampling
Task Performance
(4) Rejection-Sampling with *Reward Models* *Can* Improve Performance by filtering the generation
Prob
The effective generation distribution of an optimized LLM with rejection-sampling
Task Performance
(5) On Hard Tasks, Reward Models are Crucial as they enable search and *Inference-Time-Optimization*
Prob
The generation distribution of an iteratively optimized LLM with rejection-sampling
Task Performance
(6) Searching with Reward Models can generate datasets that enable *iterative fine-tuning*

Figure 1: *A comparison of different optimized and non-optimized usages of LLMs in general tasks.* (1) LLMs are versatile samplers that can be directly applied to many tasks, yet their performances are rarely optimal on the fly, this is because they are trained with the prior knowledge embedded in natural language corpus through n-gram autoregression. (2-3) Prompting and Fine-tuning are two effective ways of shifting the generation of LLMs to improve their performance on specific tasks. While the design of prompting is often based on heuristic designes, light-weighted, and requires a relatively small validation dataset, fine-tuning requires more demonstrative samples to effectively shift the generation distribution (4) Rejection-Sampling based on a ***learned*** reward model is another way of shifting the **effective** generation distribution of LLMs. (5) On hard tasks such as math reasoning, the insight of rejection sampling can be combined with LLM optimization techniques to further enhance the sampling efficiency. Reward models play a essential role in inference-time optimization, that can be used to enhance the inference performance. (6) Both the reward models and the LLM generators can be iteratively optimized based on the data generated in rejection-sampling. Such iterative optimization can improve the inference-time efficiency and reward modeling accuracy.

framework not only clarifies the relationships between different tasks but also facilitates the transfer of methods and insights across domains, making the research in the optimization of LLMs more transferable, cohesive, and impactful.

## 2 Optimizing the Usage of LLMs: Joint Alignment through Prompting, Fine-Tuning, and Inference-Time-Optimization

In this section, we formally introduce the notations needed for our framework.

### 2.1 Notations

We use $\mathcal{V}$ to denote the vocabulary space, and $\mathcal{V}^\infty$ to denote the natural language space. We use $\ell_{\theta,\tau}: \mathcal{V}^\infty \mapsto \mathcal{V}^\infty$ to denote LLMs parameterized by $\theta$ and generate tokens with parameter $\tau$; we use $\phi \in \mathbb{R}^{d_\mathrm{e}}$ to denote the thoughts of LLMs in embedding space — given current context as inputs, where $d_e$ is the dimension of embeddings; we use $x \in \mathcal{V}^\infty$ to denote external queries we would use LLMs to seek for an answer; We use $\mathcal{K} \subset \mathcal{V}^\infty$ to denote external knowledge, e.g., a private dataset or other external references that could be useful for specific tasks; we use $\mathcal{P}_\psi : \mathbb{R}^{d_\mathrm{e}} \times \mathcal{V}^\infty \mapsto \mathcal{V}^\infty$ to denote a prompting policy, parameterized by $\psi$. Depending on the specific settings, we may have access to an external demonstrative dataset $\mathcal{D}_\mathrm{demo} = \{x_i, y_i\}_{i=1}^N$, external preference dataset $\mathcal{D}_\mathrm{pref} = \{x_i, y_i^+, y_i^-\}_{i=1}^N$, or evaluator $\mathcal{E} : \mathcal{V}^\infty \times \mathcal{V}^\infty \mapsto \mathbb{R}$. We denote reward function $\mathcal{R} : \mathcal{V}^\infty \times \mathcal{V}^\infty \mapsto \mathbb{R}$ as a proxy of the evaluator $\mathcal{E}$.

### 2.2 Optimizing the Usage of LLMs with Alignment

With the notations above, we now introduce the underlying optimization problem in LLMs' usages:

We define the LLM $\ell_{\theta,\tau}$ as the **generator**; and the $\mathcal{P}_\psi$ as the **prompter**. In **all** LLM applications, the interactions start from an initial query $x$. Taking this initial query, the prompter $\mathcal{P}_\psi$ may take external knowledge as additional context in forming the input of LLMs. Formally, we use $t = 0, 1, 2, ..., T$ to denote the round of interactions. Moreover, we use $l = \{1, 2, ..., L\}$ to denote the fact that LLMs can generate multiple samples given a single prompt

$$\begin{aligned}
&t = 0,\ \ \phi_0 = \emptyset,\ \ r_0 = 0,\ \ p_0 = \mathcal{P}_\psi(x, \mathcal{K}, \phi_0, r_0),\\
&t = 1,\ \ \phi_1^{(l)} \sim \ell_{\theta,\tau}(p_0),\ \ r_1^{(l)} = \mathcal{R}(x, \phi_1^{(l)}),\ \ p_1 = \mathcal{P}_\psi(x, \mathcal{K}, \phi_1^{(l)}, r_1^{(l)}),\\
&...\\
&t = T,\ \ \phi_T^{(l)} = \ell_{\theta,\tau}(p_{T-1}),\ \ r_T^{(l)} = \mathcal{R}(x, \phi_T^{(l)}),
\end{aligned} \tag{1}$$

The underlying objective of using LLMs for different tasks is always to maximize the evaluation performance of the final LLM outputs (e.g., inference-time performance). Since we do not have access to the inference-time evaluator, reward models are utilized as an optimization proxy (e.g., training-set performance). Therefore, the objective is to find

$$\theta^*, \psi^*, l^* = \arg\max_{\theta,\psi,l} \mathcal{J}(\theta, \psi, l) = \arg \max_{\theta,\psi,l\in\{1,2,...,L\}} \mathbb{E}_{x,p_t\sim\mathcal{P}_\psi,\phi_t\sim\ell_{\theta,\tau}} \mathcal{R}(x, \phi_T^{(l)}), \tag{2}$$

such that

$$\mathbb{E}_{x,p_t\sim\mathcal{P}_{\psi^*},\phi_t\sim\ell_{\theta^*,\tau}} \mathcal{E}(x, \phi_T^{(l^*)}) = \max_{\theta,\psi,l\in\{1,2,...,L\}} \mathbb{E}_{x,p_t\sim\mathcal{P}_\psi,\phi_t\sim\ell_{\theta,\tau}} \mathcal{E}(x, \phi_T^{(l)}), \forall\theta, \psi, l \tag{3}$$

The overall objective is to improve the task performance using the evaluator. The reward function is used as a proxy to guide the generation process to maximize the evaluator. Therefore, we need (1) maximize $\mathcal{R}$, and (2) align $\mathcal{E}$ and $\mathcal{R}$ such that we are optimizing toward the correct objective.

## 3 The Forward Problem: The Critical Role of Reward in Optimization

With a known reward function, the shared insight behind LLMs' applications in different tasks is rejection sampling [1]. In such a process, the two players [2] act as **generators** of diverse responses, and an evaluation metric is given to guide the selection of responses. Overall, there are three orthogonal classes of optimizing the objective function in Eqn.(2):

1. Optimizing $\theta$: Corresponds to fine-tuning language models
2. Optimizing $\psi$: Corresponds to prompt optimization in generation
3. Increasing the number of $L$: Corresponds to increasing the probability of hitting high-rewarded regions in the response space.

### 3.1 LLMs for Reasoning Tasks

The evaluator $\mathcal{E}$ in those tasks are indicator functions, judging whether the answers are correct.

#### 3.1.1 Methods without Reward Models

**Supervised Fine-Tuning for Reasoning: Optimization over $\theta$** For any task with an expert demonstration dataset, the most straightforward way of improving LLMs' performance on the task is to conduct supervised fine-tuning — such that we may expect the LLMs' problem-solving ability in the test time to improve. In the reasoning task, $T = 1, L = 1, \mathcal{K} = \emptyset$, and $\mathcal{P}_\psi(x) = x$.

**Chain of Thought Prompting: A (Non-optimised) Heuristic $\psi$** In zero-shot chain-of-thought prompting [3], $T = 1, L = 1$, $\mathcal{K} = \emptyset$, $\mathcal{P}_\psi(x) = x \oplus$ `Let's think step by step`, where $\oplus$ denotes concatenation.

**In Context Learning: An Optimized Heuristic $\psi$** In few-shot chain-of-thought prompting [4], $T = 1, L = 1, \mathcal{K} = \mathcal{D}_{\text{demo}}, \mathcal{P}_\psi(x, \mathcal{K}) = \bigoplus_{m=1}^{M} K_m(x) \oplus x$, where we use $\bigoplus$ to denote aggregation of multiple demonstrative examples, and $K_m(x) \in \mathcal{K}$ are the selected demonstrative examples related to $x$.

### 3.1.2 Methods with Reward Models

**Thought Processes: Optimization over and $L$ with (Non-optimised) Heuristic $\psi$** In thought processes, e.g., the Tree-of-Thought[5], Graph-of-Thought[6], and RATP [7], $T > 1, L > 1$, $\mathcal{K}$ is an optional private database, $\mathcal{P}_\psi$ is a pre-defined template that stimulates the LLM to generate diverse thoughts. The reward models $\mathcal{R}$ in those methods are always implemented by LLM Self-Critics.

**Prompt Optimization for Reasoning: Optimization over $\psi$** There is literature working on improving the prompting policy such that the reasoning performance can be improved [8]. In Prompt-OIRL, $T = 1, L > 1, \mathcal{K} = \emptyset$, and the $\mathcal{P}_\psi$ selected the highest rewarded prompt. The reward model $\mathcal{R}$ is selected to be tree-based classifiers.

## 3.2 LLMs for Evaluation

**Building Reward Models in RLHF: Optimization over $\theta$** In RLHF, reward models are crucial in supervising the policy learning process for specific tasks. Those reward models are usually instantiated by using a language model with a value head or generative language models. In both cases, $T = 1, L = 1, \mathcal{K} = \emptyset, \mathcal{P}_\psi(x) = x$, and the reward models are optimized to predict the correct labels through supervised learning.

**LLMs as Critics: A (Non-optimised) Heuristic $\psi$** LLMs are introduced to act as critics or reward models that evaluate natural language contents (often generated by other LLMs). In those use cases, $T = 1, L = 1, \mathcal{K} = \emptyset$, $\mathcal{P}_\psi$ are pre-defined prompts that compare multiple contents or provide evaluation scores.

## 3.3 LLMs for External System Optimization

In those tasks, the reward function $\mathcal{R}$ is instantiated as an external system that can provide feedback. e.g., a robotics control environment. And the reward values are generated through those external systems.

**LLMs for Reward Design in Robotics: Optimization over $L$ with a Heuristic $\psi$** In EUREKA [9], LLMs are used to generate reward functions for control systems. $T > 1, L = 1, \mathcal{K} = \emptyset, \mathcal{P}_\psi$ is a heuristically designed prompting template that motivates the LLM to generate potential reward function candidates. The $\mathcal{R}$ is given by external systems, i.e., whether the control task is successfully completed.

**LLMs for Bayesian Optimization: Optimization over $L$ with a Heuristic $\psi$** In LLaMBO [10], LLMs are used to enhance Bayesian Optimization (BO). In such a process, the performance of BO methods are leveraged as external reward signal, and $T > 1, L > 1, \mathcal{K} = \emptyset, \mathcal{P}_\psi$ is a heuristically designed prompting template that motivates the LLM to generate diverse BO hyperparameters.

Table 1: *Comparison of different optimization strategies in LLM usage.* ITO: Inference-Time Optimization; ● denotes test time optimization through interaction with external environment they are tested on.

| **Task** | **LLM Usage** | $\theta$ | $\mathcal{P}_\psi$ | $T$ | $L$ | $\mathcal{K}$ | $\mathcal{R}$ | ITO |
|---|---|---|---|---|---|---|---|---|
| General (incl. Chat) | Supervised Fine-Tuning | **Optimized** | Fixed | 1 | 1 | $\emptyset$ | NA | ✗ |
| | InverseRLignment | **Optimized** | Fixed | 1 | $> 1$ | $\emptyset$ | IRL-RM | ✓ |
| | RLHF | **Optimized** | Fixed | 1 | $> 1$ | $\emptyset$ | BT-RM | ✓ |
| Reasoning | Zero-Shot Prompting | Fixed | Heuristic | 1 | 1 | $\emptyset$ | NA | ✗ |
| | Few-Shot Prompting (ICL) | Fixed | Heuristic | 1 | 1 | $\neq \emptyset$ | NA | – |
| | Thought Processes | Fixed | Heuristic | $> 1$ | $> 1$ | $\neq \emptyset$ | LLM-Critic | ✓ |
| | Prompt Optimization | Fixed | **Optimized** | 1 | $> 1$ | $\emptyset$ | Binary-RM | ✓ |
| Evaluation | Reward Models in RLHF | **Optimized** | Fixed | 1 | 1 | $\emptyset$ | NA | ✗ |
| | LLMs for Evaluation | Fixed | Heuristic | 1 | 1 | $\emptyset$ | NA | ✗ |
| External Sys. Optimization | LLMs for Robotics | Fixed | Heuristic | $> 1$ | $> 1$ | $\emptyset$ | External | ● |
| | LLMs for BO | Fixed | Heuristic | $> 1$ | $> 1$ | $\emptyset$ | External | ● |

**Insights:** Based on the comparison table and the comparisons above, we find:

1. Reward Models play a vital role in optimizing the usage of LLMs, without a reward model, it is impossible to perform inference-time optimization. While LLM self-critic can serve as a heuristic proxy for such a reward model, their performance is not guaranteed as they are not optimized toward the specific judging task. Building reward models from data is necessary.
2. The usage of LLMs is largely sub-optimal. For the pursuit of the optimal usage of LLMs, all three components: the Language Model $\theta$, the prompter $\mathcal{P}_\psi$, and inference-time optimization should be jointly optimized.
3. As an important special case, we note that while LLMs are widely used for evaluation tasks, their usages are not yet optimized. Conceptually, there is no difference between using LLMs for evaluation and using LLMs for reasoning with zero-shot prompting. Any usage of LLMs — versatile samplers — could be improved by considering the alternative optimization procedures such as prompt optimization, model fine-tuning, and rejection sampling with a reward model, or a combination of them.

## 4 The Inverse Problem: Building Reward Models from Data

As discussed in the previous section, providing LLMs with feedback is crucial for enhancing their performance across general tasks — either through a learned reward model that is a proxy for an inaccessible task evaluator or an external reward signal when the task evaluator is available. However, in practice, an external reward signal is not always accessible, necessitating the construction of reward models from datasets. In this section, we focus on the **Inverse Problem** of alignment, exploring existing techniques and approaches for generating reward models from datasets.

### 4.1 Building Reward Models from Preference Annotations (e.g., RLHF)

One of the most widely recognized approaches for building reward models in LLM alignment is through reinforcement learning from human feedback (RLHF). In this learning paradigm, obtaining direct scalar feedback from humans is often infeasible, leading to the adoption of preference-based learning. Formally, the dataset in RLHF — or in general preference-based learning — takes the form of $\mathcal{D}_{\text{pref}} = \{x_i, y_i^+, y_i^-\}_{i=1}^N$, where $x_i$ is the prompt, $y_i^+$ is the preferred language model response and $y_i^-$ the dispreferred model response. This structure allows the model to learn from pairwise comparisons, effectively capturing human preferences and aligning the model's outputs with desired outcomes.

In practice, RLHF is particularly valuable in scenarios where generating expert demonstrations is challenging, and no gold-standard reward model or evaluation metric exists to accurately assess the quality of different responses. This is often the case in tasks such as summarization, reducing chatbot harmfulness, and enhancing chatbot helpfulness, where subjective judgments are required.

Technically, a common approach for reward modeling using pairwise preference annotations is to employ the Bradley-Terry model. This model is used to optimize the following objective:

$$\mathcal{L}(\omega) = -\sum_{i=1}^{N} \log \frac{\exp(r_\omega(y_i^+|x))}{\exp(r_\omega(y_i^+|x)) + \exp(r_\omega(y_i^-|x))}, \tag{4}$$

where $r_\omega(y|x)$ is the reward model — parameterized by $\omega$ — that evaluates response $y$ given prompt $x$. The objective function maximizes the likelihood that the model assigns a higher score to the preferred response $y_i^+$ over the dispreferred response $y_i^-$. This approach effectively trains the model to align its outputs with human preferences in the absence of explicit reward signals or definitive evaluation metrics.

In RLHF literature, the reward models are trained to be aligned with the binary human preferences [11], therefore it is naturally a sparse reward in the sence that it allocates a reward value for the entire response generation. Recent work on dense reward models [12, 13] highlighted the superiority of a token-level reward model, as complementary of the empirical justification [14].

While the majority of advancements in RLHF [15–20] rely on preference-based datasets annotated by humans or general-purpose LLMs [21–23], several significant challenges such as labeling noise [24, 19], privacy issue [25, 7], and high cost [1, 23, 26, 27] impede their performance and limit their applications. To address those challenges, recent advances in LLM alignment [28, 29] introduced the demonstration-based alignment techniques [30].

In literature, there are two broad classes of tasks that reward models can be built based on demonstration datasets, namely (1) the tasks without an accurate quality measure — such as the chat task when the objective is to reduce the toxicity and improve helpfulness, and (2) the tasks with an accurate quality metric — such as mathematical reasoning and coding. In the following, we elaborate on how to build reward models for those different classes of tasks.

### 4.2 Building Reward Models from Expert Demonstrations without a Golden Metric

In [28], the authors introduce an adversarial imitation learning-based method to build reward models from demonstration datasets for chat alignment tasks. A similar idea has been implicitly explored in the self-play based methods [31] where the direct preference optimization method [32] is applied to the synthetic pairwise dataset, which is augmented from the expert demonstrations with the current model's generation as its negative samples.

The shared insight behind those methods is the adversarial distributional matching [33–35], where a discriminative model is used to optimize the generation process such that it is non-distinguishable from the expert demonstrations. In [28], the discriminative model is introduced as a reward model through extrapolation. Formally, with the dataset under the format of $\mathcal{D}_{\text{demo}} = \{x_i, y_i^*\}$, where $x_i$ denotes the prompt and $y_i^*$ the expert response, a practical policy learning objective is given as

$$\max_{\pi} \mathbb{E}_{(y|x)\sim\pi} \left[\log D_\omega(y|x) - \log(1 - D_\omega(y|x))\right]. \tag{5}$$

The discriminative mode $D_\omega$ can be optimized through:

$$\max_{\omega} \mathbb{E}_{(y|x)\sim\mathcal{D}_{\text{demo}}}[\log D_\omega(y|x)] + \mathbb{E}_{(y|x)\sim\pi}[\log(1 - D_\omega(y|x))]. \tag{6}$$

Using the reward notion

$$r_\omega(y|x) = \log D_\omega(y|x) - \log(1 - D_\omega(y|x)), \tag{7}$$

when $D_\omega(y|x)$ is instantiated by neural networks with sigmoid activation function over logits $D_\omega(y|x) = \sigma(\texttt{logits}(y|x,\omega))$, we have $r_\omega(y|x) = \texttt{logits}(y|x,\omega)$.

### 4.3 Building Reward Models from Expert Demonstrations with a Golden Metric

Besides the tasks where golden metric is hard to define, there are lots of tasks ther golden metric is known and easily accessible, such as boolean question answering [36], mathematical reasoning [37–39], and code generation [40]. In those tasks, the LLM-generated answers are easily verifiable — we can easily compare the derived answer (or by comparing outputs of programs) to the golden answers and quantitatively evaluate the performance of the LLMs on each individual questions.

In those tasks, training the Outcome-supervised Reward Models (ORMs) [39, 41] is the most straightforward approach, and it does not require extra annotations in addition to the desired outcomes in the training dataset. Formally, when giving a demonstration dataset $\mathcal{D}_{\text{demo}} = \{x_i, y_i^*\}$, where $x_i$ denotes the prompt and $y_i^*$ the corresponding golden answer, for any natural language response $y_i$ generated by LLMs, the learning objective of the ORMs is to optimize the $\omega$-parameterized reward model $R_\omega$

$$\mathcal{L}(\omega) = -\mathbb{E}_{i|\mathcal{E}(y_i,y_i^*)=1}\left[\log R_\omega(x_i, y_i)\right] + \mathbb{E}_{i|\mathcal{E}(y_i,y_i^*)=0}\left[\log(1 - R_\omega(x_i, y_i))\right] \tag{8}$$

Here, $\mathcal{E}(y_i, y_i^*)$ is a binary function such that:

$$\mathcal{E}(y_i, y_i^*) = \begin{cases} 1, & \text{if } y_i \text{ is the correct answer} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

**Dense Reward and Search-based Generation** To effectively solve complex tasks, researchers have introduced various thought process models including chain-of-thought [4], tree-of-thought [5], graph-of-thought [6], and more general thought processes [7]. These approaches enable LLMs to engage in think-aloud processes [42, 43], mirroring human problem-solving strategies. However, in Equation (1) of Section 2.2, if the reward function is as described, we encounter a sparse reward problem: the model receives a binary feedback (1 for correct, 0 for incorrect) only at the end of the process. This leads to the well-known credit assignment challenge in temporal-based learning [44, 45]. To address this limitation, Process-supervised Reward Models (PRMs) [46] propose using additional annotations at each step of the thought process, creating a more fine-grained reward model. Building on this concept, [47] introduces a tree-based searching method to generate process rewards without relying on external annotators. The overarching principle of utilizing dense rewards [12, 13] across various tasks is to provide detailed feedback throughout the LLM generation process, enabling better control through search-based decoding [48, 49]. Generally, dense rewards – which don't necessarily need to be as granular as token-level feedback – can guide search processes and have the potential to be combined with advanced techniques like Monte Carlo Tree Search with value estimators [50], further enhancing the capabilities of LLMs in complex problem-solving scenarios.

**Iterative Optimization with Search-Based Data Augmentation** To construct more precise reward models, particularly for tasks involving multi-step responses, search-based augmentation can be employed to expand the original dataset and enhance the reward models' generalization capabilities. For instance, in mathematical reasoning and Boolean question answering with external databases [47, 7], diverse reasoning steps are generated via search algorithms. The validity of these steps can then be assessed based on the final outcomes they produce, often utilizing techniques such as Monte Carlo estimation. This augmentation process transforms the original dataset, which typically comprises questions and their corresponding golden answers, into a richer reasoning path dataset. This enhanced dataset encompasses questions, various reasoning steps, and indicators of the correctness of answers derived from these steps. Such a comprehensive dataset facilitates credit assignment learning and enables a more nuanced evaluation of each reasoning step's value. Moreover, the correctly identified reasoning paths can serve as a valuable supervised fine-tuning dataset, directly enhancing the model's reasoning capabilities. This iterative approach not only improves the reward model's accuracy but also bolsters the LLM's ability to generate coherent and logically sound multi-step responses across a broader range of tasks.

**Insights:** From the reviewed approaches, we observe:

1. Building reward models from data is essential for aligning LLM outputs with human preferences, especially when explicit reward signals are unavailable. Pairwise preference annotations, like in RLHF, are widely used, but dense, token-level rewards provide finer control in complex tasks.
2. Demonstration-based methods, especially for tasks without clear metrics (e.g., chatbots), leverage adversarial learning to align models with expert responses, while outcome-supervised models suit tasks with verifiable metrics (e.g., coding).
3. Search-based augmentation and dense rewards help solve multi-step tasks by offering fine-grained feedback, improving reasoning capabilities and optimizing the generation process. Data augmentation further enriches datasets with reasoning steps, enhancing model generalization and credit assignment learning.

## 5 Conclusive Remark

This research presents a unified optimization framework for Large Language Models (LLMs), integrating various enhancement methods previously studied in isolation. The framework emphasizes three key insights: the need for alignment beyond basic sampling capabilities for true task mastery, the crucial role of reward modeling in inference-time optimization, and the importance of choosing appropriate reward models based on task properties and dataset availability. This holistic approach provides a systematic method for analyzing existing LLM optimization techniques and identifies strategies to maximize LLM potential across diverse applications. The framework underscores the importance of joint optimization of the language model, prompting strategies, and inference-time techniques to achieve optimal performance. It also highlights the potential of iterative optimization with search-based data augmentation for complex multi-step tasks, which can enhance reward model accuracy and improve LLM generalization capabilities. By bridging the gap between different optimization strategies and emphasizing the critical role of reward modeling, this research provides a foundation for future advancements in LLM alignment and optimization, potentially leading to more capable, reliable, and task-specific language models.

## References

[1] Wei Xiong, Hanze Dong, Chenlu Ye, Han Zhong, Nan Jiang, and Tong Zhang. Gibbs sampling from human feedback: A provable kl-constrained framework for rlhf. *arXiv preprint arXiv:2312.11456*, 2023.

[2] Rui Zheng, Hongyi Guo, Zhihan Liu, Xiaoying Zhang, Yuanshun Yao, Xiaojun Xu, Zhaoran Wang, Zhiheng Xi, Tao Gui, Qi Zhang, et al. Toward optimal llm alignments using two-player games. *arXiv preprint arXiv:2406.10977*, 2024.

[3] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[4] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[5] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690, 2024.

[7] Thomas Pouplin, Hao Sun, Samuel Holt, and Mihaela Van der Schaar. Retrieval-augmented thought process as sequential decision making. *arXiv preprint arXiv:2402.07812*, 2024.

[8] Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse rl. In *The Twelfth International Conference on Learning Representations*, 2023.

[9] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[10] Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024.

[11] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[12] Alex J Chan, Hao Sun, Samuel Holt, and Mihaela van der Schaar. Dense reward for free in reinforcement learning from human feedback. *arXiv preprint arXiv:2402.00782*, 2024.

[13] Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From $r$ to $q$: Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024.

[14] Ziniu Li, Tian Xu, Yushun Zhang, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

[15] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[16] Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*, 2023.

[17] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[18] Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

[19] Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*, 2023.

[20] Rémi Munos, Michal Valko, Daniele Calandriello, Mohammad Gheshlaghi Azar, Mark Rowland, Zhaohan Daniel Guo, Yunhao Tang, Matthieu Geist, Thomas Mesnard, Andrea Michi, et al. Nash learning from human feedback. *arXiv preprint arXiv:2312.00886*, 2023.

[21] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[22] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

[23] Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, et al. Direct language model alignment from online ai feedback. *arXiv preprint arXiv:2402.04792*, 2024.

[24] Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

[25] Haoran Li, Yulin Chen, Jinglong Luo, Yan Kang, Xiaojin Zhang, Qi Hu, Chunkit Chan, and Yangqiu Song. Privacy in large language models: Attacks, defenses and future directions. *arXiv preprint arXiv:2310.10383*, 2023.

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[27] Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, et al. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, 2024.

[28] Hao Sun and Mihaela van der Schaar. Inverse-rlignment: Inverse reinforcement learning from demonstrations for llm alignment. *arXiv preprint arXiv:2405.15624*, 2024.

[29] Omar Shaikh, Michelle Lam, Joey Hejna, Yijia Shao, Michael Bernstein, and Diyi Yang. Show, don't tell: Aligning language models with demonstrated feedback. *arXiv preprint arXiv:2406.00888*, 2024.

[30] Zhihan Liu, Yufeng Zhang, Zuyue Fu, Zhuoran Yang, and Zhaoran Wang. Learning from demonstration: Provably efficient adversarial policy imitation with linear function approximation. In *International conference on machine learning*, pages 14094–14138. PMLR, 2022.

[31] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

[32] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[33] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[34] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[35] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[36] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[37] Subhro Roy and Dan Roth. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*, 2016.

[38] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.

[39] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[40] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[41] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[42] Maarten Van Someren, Yvonne F Barnard, and J Sandberg. The think aloud method: a practical approach to modelling cognitive. *London: AcademicPress*, 11(6), 1994.

[43] Gregor Betz, Kyle Richardson, and Christian Voigt. Thinking aloud: Dynamic context generation improves zero-shot reasoning performance of gpt-2. *arXiv preprint arXiv:2103.13033*, 2021.

[44] Thomas Mesnard, Wenqi Chen, Alaa Saade, Yunhao Tang, Mark Rowland, Theophane Weber, Clare Lyle, Audrunas Gruslys, Michal Valko, Will Dabney, et al. Quantile credit assignment. In *International Conference on Machine Learning*, pages 24517–24531. PMLR, 2023.

[45] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

[46] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[47] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023.

[48] James Y Huang, Sailik Sengupta, Daniele Bonadiman, Yi-an Lai, Arshit Gupta, Nikolaos Pappas, Saab Mansour, Katrin Kirchoff, and Dan Roth. Deal: Decoding-time alignment for large language models. *arXiv preprint arXiv:2402.06147*, 2024.

[49] Ruizhe Shi, Yifang Chen, Yushi Hu, ALisa Liu, Noah Smith, Hannaneh Hajishirzi, and Simon Du. Decoding-time language model alignment with multiple objectives. *arXiv preprint arXiv:2406.18853*, 2024.

[50] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.