Aligning Large Language Models with Human Feedback: Mathematical Foundations and Algorithm Design

Chenliang Li*, Jiaxiang Li*, Luca Viano*, Siliang Zeng*,
Volkan Cevher, Markus Wulfmeier, Stefano Ermon, Alfredo Garcia, Mingyi Hong

Abstract

This article provides an introduction to the mathematical foundations and algorithmic frameworks used to align Large Language Models (LLMs) with human intentions, preferences, and values. We discuss standard alignment techniques, such as fine-tuning (SFT), reinforcement learning with human feedback (RLHF), and direct preference optimization (DPO). We also explore the theoretical underpinnings of learning from human preferences, drawing connections to inverse reinforcement learning (IRL) and discrete choice models. We present state-of-the-art algorithms in a tutorial style, discuss their advantages and limitations, and offer insights into practical implementation. Our exposition is intended to serve as a comprehensive resource for researchers and practitioners, providing both a foundational understanding of alignment methodologies and a framework for developing more robust and scalable alignment techniques.

I. Introduction

Background. As large language models (LLMs) have taken the world by storm, it is clear that generative AI systems will soon become ubiquitous in our lives. LLMs have been applied beyond chatbots and personal assistants to tackle complex challenges, including video gaming [1] and autonomous control [2]. In this context, the concept of *alignment* plays an increasingly important role in the design and training of AI systems. Loosely speaking, alignment refers to the performance guarantee that the AI system will generate outcomes that are intended or preferred by the human user without undesirable side effects or behaviors such as deception or manipulation.

More technically, the LLM alignment problem involves fine-tuning or adjusting a base model that was originally trained on extensive, diverse datasets. While such a base model is trained to learn core behaviors and general knowledge, it often falls short when applied to specific tasks or to meeting the nuanced preferences of a human user. To better *align* the model, additional feedback is gathered from human experts

*: equal contribution, ordered alphabetically.

who evaluate and/or rank the model's performance within a specific task context. This feedback embodies expert judgments regarding both the accuracy of the model's responses and any particular preferences they may have. The goal is to refine the model to better meet these specific requirements, ultimately enhancing its task relevance and user satisfaction. Generally, it is observed that after the alignment process, LLMs can follow human instructions well, and can avoid providing toxic and non-preferable responses. Refer to Fig. 1 as an illustration of an overview of the alignment process.

Contributions of This Work. Despite extensive study of the alignment problem in recent literature, this research area remains young, and it is still evolving rapidly with many unresolved theoretical challenges. A primary contribution of this article is to introduce the signal processing community to the fundamentals of alignment, highlight state-of-the-art algorithms and approaches, and clarify the intersection between alignment methodologies and various techniques relevant to signal processing (SP). We believe that our article will foster interdisciplinary contributions and expand the role of SP in LLM alignment research. By understanding the alignment problem, SP researchers can leverage their expertise in mathematical modeling, noise handling, inverse problem-solving, and optimization to address a number of key challenges. For instance, similar to reconstructing signals from observed data in SP, many alignment approaches involve solving inverse problems, such as learning the human preference models from the data. Therefore, we expect that introducing the alignment to SP professionals can foster innovation in AI safety, inspire novel techniques, and accelerate the development of robust, generalizable alignment frameworks that can be applied across diverse applications.

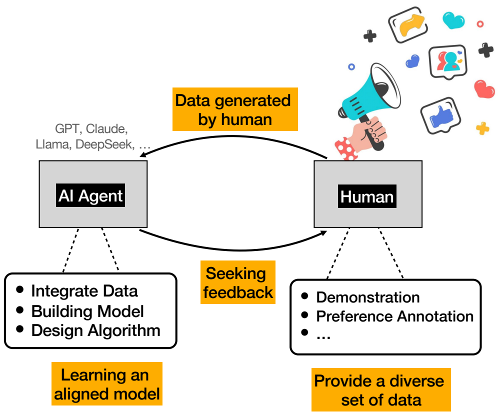


Fig. 1: An illustration of the alignment process.

Organization of this work. The article is organized as follows. In Sec. II, we provide the readers with an overview of the LLM training process. In Sec. III, we discuss some classical notion and techniques of learning from human preference, which serves as the foundation for many alignment approaches to be discussed subsequently. In Sec. IV, we provide a detailed discussion of a number of state-of-the-art algorithms for different stages of LLM alignment. In Sec. V, we provide readers with a framework that unify a few alignment algorithms discussed in the previous section under a single formulation. In Sec. VI, we provide some high-level discussion In Sec. VII, we conclude with final remarks and open questions.

II. Preliminaries

In this section, we provide a basic description of the LLM training process, while emphasizing the basic concepts relevant for alignment problems that we describe in the sequel.

LLM Representation. Consider an LLM parameterized by parameters θ and denote the output probability by $\pi(y|x;\theta)$ where $x := [x_1, \ldots, x_n]$ is the sequence of input prompts and $y := [y_1, \ldots, y_m]$ is the sequence of output continuation, where n and m are their sequence length, correspondingly. In practice, each one of y_i and x_j is a token, which is one element of a token vocabulary. A token is a discrete unit that represents certain semantic information. For example, a word "learning" itself could be one token in the vocabulary, however more commonly two elements "learn" and "-ing" are both tokens where "-ing" represents the semantic information of tense. A tokenizer is commonly utilized to transform an input natural language sentence into sequence of tokens $x = [x_1, \ldots, x_n]$.

Typical LLM is an auto-regressive model, meaning that it predicts the output probability of the y_j given all tokens in x and $y_{<j} := [y_1, \ldots, y_{j-1}]$ ($y_{<1}$ is null). More precisely, the probability of producing a sequence y can be defined as:

$$\pi(y|x;\theta) = \prod_{j=1}^{m} \pi(y_j|x, y_{<j};\theta). \quad (1)$$

The state-of-the-art LLMs are built upon the attention mechanism [3] and θ represents all the trainable parameters, including the attention matrices and weights of the feed-forward layers.

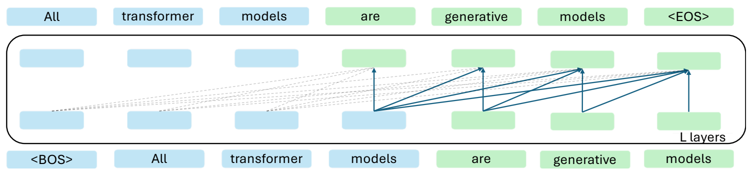


Fig. 2: An illustration of Auto-regressive LLMs. Here <BOS> and <EOS> are two special tokens representing the begin and the end of the sentence, respectively. The tokens at the bottom are the input token sequence, and those at the top are the output tokens. In this example, the token sequence "<BOS> All transformer models " is given to the LLM and "are generative models <EOS>" is the prediction. Each of the previous tokens will be used to predict the next token, corresponding to (1).

Next, let us provide an introduction about different stages of LLM training.

The LLM pre-training. The first stage of the training is called *pre-training*, where an LLM model is initialized randomly and trained over a vast corpus of documents, known as the pre-training dataset. As an example, Llama 3 model is pre-trained on more than 15T token of documents collected from all over the Internet, with the entire token vocabulary size (i.e., the total number of distinct tokens) of 128K [4]. The pre-training dataset is an unsupervised dataset, denoted as $\mathcal{D}_{\text{pre}} := \{x^i\}_{i=1}^{N_{\text{pre}}}$, where each data $x^i = [x^i_1, ..., x^i_{m_i}]$ is a collection of tokens with m_i being its sequence length. $\mathcal{D}$ is a dataset without labels usually collected from the internet, such as Wikipedia articles; the learning task at the pre-training stage is typically formulated as the so-called *next-token prediction* task, which predicts the next token given the current and all previous tokens shown to the LLM, and can be mathematically formulated as a negative log-likelihood minimization problem:

$$\min_{\theta} -\frac{1}{N_{\text{pre}}}\sum_{i=1}^{N_{\text{pre}}}\log\pi(x^i;\theta) = -\frac{1}{N_{\text{pre}}}\sum_{i=1}^{N_{\text{pre}}}\sum_{j=1}^{m_i}\log\pi(x^i_j|x^i_{<j};\theta). \tag{2}$$

This task is commonly understood as encouraging the LLM to memorize all the given texts and enabling the LLM for more fine-grained tasks.

The LLM Alignment. Once the pre-training is done, the next step is called *alignment* or *fine-tuning*. The fine-tuning process is a supervised process that aims at improving the instruction following capabilities of LLMs to better align with human behaviors and values. It usually consists of two main steps: the supervised fine-tuning (SFT) step and the reinforcement learning with human feedback (RLHF) step.

The SFT step utilizes a demonstration dataset $\mathcal{D}_{\text{demo}} := \{(x_i, y_i)\}_{i=1}^{N_{\text{demo}}}$, where x is the input prompt (such as, “Where is the capital of the US?”) and y is the expert response (such as, “Washington, DC”). We assume that the demonstration continuations y are collected from an *expert* (typically a human expert, but sometimes it can also be a very large and well-aligned model such as GPT4 which produces high quality responses), thus we also denote $(x, y) \sim \mathcal{D}_{\text{demo}}$ as $x \sim \rho, y \sim \pi^E(\cdot|x)$ as their population distributions, where ρ is the distribution of the input prompts when collecting the data, and π^E is the expert policy.

Meanwhile the RLHF step first utilizes a *preference dataset* to learn from human preference about the quality of the answers to a particular question x. This dataset is denoted as $\mathcal{D}_{\text{pref}} := \{(x, y_w \succ y_l)\}$, where y_w is preferred (i.e., ‘$\underline{\text{w}}$ins’) over y_l (i.e., ‘$\underline{\text{l}}$oses’) by a human labeler, and will be succinctly denoted as $(y_w \succ y_l)$ throughout the paper. We use the notation $x \sim \rho, (y_w \succ y_l) \sim \pi^P(\cdot|x)$ to denote the population distribution of the preference dataset, where π^P is the preference distribution. The preference data is typically used to learn a parameterized *reward function* $r(x, y; \phi)$, which provides a score for a given prompt-response pair (x, y). Once such a reward function is learned, the RLHF step leverages this reward function, together with a *prompt dataset* $\mathcal{D}_{\text{prompt}} := \{x\}$, to evaluate and further improve the quality of the generated responses, through a reinforcement learning (RL) process. The whole process of RLHF is summarized and illustrated in Fig. 3.

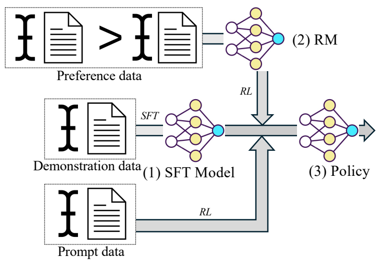


Fig. 3: An illustration of the standard LLM Alignment process.

The following example illustrates the demonstration and the preference data.

Example 1: LLM Alignment Data. Given a question prompt "How should I respond to an email from my professor asking for a delayed assignment?" The demonstration data includes a *preferred* answer "You can politely acknowledge the delay and provide a reasonable explanation. For example: 'Dear Professor, I apologize for the delay in submitting my assignment. I encountered unexpected challenges but will submit it by [new deadline]. Thank you for your understanding." Meanwhile, the preference dataset includes an additional *non-preferred* answer, e.g., "Just say you were busy and didn't have time. Professors have to deal with it." Clearly, the preferred response aligns with human values of politeness and professionalism.

We will provide a detailed account of popular methods to learn from these different kinds of datasets in the subsequent sections. In a high level, the SFT step can be used to directly teach LLM to imitate the expert behavior, while the RLHF step teaches LLM to distinguish the better behavior over the worse ones, in order to achieve a better generalization.

Based on the above discussion, we can now define the LLM alignment problem as follows:

LLM alignment problem: Given a diverse dataset representing human demonstrations and preferences, how can we effectively train an LLM to consistently generate outputs that align with human values, ethical principles, and intended use cases?

The LLM Evaluation. Before closing this introductory section, let us briefly discuss how the LLMs are evaluated after different stages of training. At the pre-training stage, the model learns to predict the next token probabilistically. The model is thus evaluated on how well it predicts the next token, and the common measure is the perplexity (PPL). Mathematically, perplexity is defined as the exponential of the average log-likelihood of the predicted words in a sequence (i.e. the exponential of the loss (2)). Hence, a lower perplexity indicates a clearer prediction and less uncertainty of the prediction of the next tokens.

In the fine-tuning stage, multiple metrics are typically used. After the SFT step, the metric is usually determined by the nature of the task. For classification tasks such as sentiment analysis (i.e., given a text of words, determine whether the sentiment of the text is positive or negative), traditional metrics, such as accuracy, precision and recall are used; For translation tasks, BLEU (BiLingual Evaluation Understudy) score is the most popular metric, which measures how similar a machine-translated text is to a human-written reference translation. After the RLHF step, the aligned model could be evaluated by win-rates from human raters or some judge models (typically takes the form of a specialized LLM), which calculate the favorability of the model response over other models/ground truth response.

In the next section, we will begin our technical discussion by delving into classical techniques on learning from human feedback data. This will serve as mathematical foundation for various alignment techinques to be discuss subsequently.

III. Theoretical Foundations of Learning from Human Feedback.

Given the background above, it is clear that the alignment problem is closely related to the problem of modeling human behaviors through the observed choices. Indeed, in the context of LLM, the response y in the demonstration data as well as the labels that distinguish y_w's with y_l's can both be viewed as human choices. It turns out that there is a vast literature about developing models for discrete choices, and these models were first used to describe discrimination between perceptual stimuli [5] and choices for urban transportation modes [6]. In this section, we will provide a brief overview of this literature.

A. Classical models of discrete choice

The choice models are used to describe observed choices (or actions) $a \in A$ made by a decision maker given a relevant state variable $s \in S$. A parallel to LLM alignment problem described in the previous section can be made by having the state s correspond to the prompt x and the action a corresponds to the possible responses y's to the prompt.

As in the previous section, we use the notation $(s,a) \succ (s,a')$ to indicate that in state s action $a \in A$ is preferred over $a' \in A$. An assumed *structure* of preferences over action set A is a relation $\succ$, which is

- *Complete*: $(s,a) \succ (s,a')$ or $a' \succ a$ for all $a, a' \in A$ and $s \in S$.
- *Transitive*: $(s,a) \succ (s,a')$ and $(s,a') \succ (s,a'')$ implies $(s,a) \succ (s,a'')$, for all $a, a', a'' \in A$, $s \in S$.

A structure of preferences can be modeled by a reward (or utility) function $r : S \times A \mapsto \mathbb{R}$, such that $(s,a) \succ (s,a')$ if and only if $r(s,a) > r(s,a')$. This implies when in state s, the decision maker will always prefer a to a'. However in practice, one may observe a decision maker engages in *mixing*, i.e. implementing different actions when in the same state $s \in S$. In the context of LLM alignment, mixing corresponds to observing different responses to the same prompt.

There are several ways of modeling mixing in the literature of discrete choice:

Random Utility: In this model, the decision maker perceives a random reward (or utility) $r(s,a) + \epsilon(a)$ (where $\epsilon(a)$ is a random variable) given the pair (s,a). A pair (s,a) is preferred to (s,a') if:

$$r(s,a) + \epsilon(a) > r(s,a') + \epsilon(a').$$

For a given state $s \in \mathcal{S}$ and a reward $r(\cdot)$, the probability of such preference can be modeled as follows:

$$P_r\Big((s,a) \succ (s,a')\Big) := \text{Prob}\Big(r(s,a) + \epsilon(a) > r(s,a') + \epsilon(a')\Big) = \text{Prob}\Big(\epsilon(a) - \epsilon(a') > r(s,a') - r(s,a)\Big).$$

When the random variables $\{\epsilon(a) : a \in A\}$ are assumed i.i.d and standard Gumbel we obtain the following logit model [7], [8] (also known as the Bradley-Terry-Luce model (BTL)):

$$P_r\Big((s,a) \succ (s,a')\Big) = \frac{\exp r(s,a)}{\exp r(s,a) + \exp r(s,a')} = \sigma\Big(r(s,a) - r(s,a')\Big), \tag{3}$$

where $\sigma(x) := \frac{1}{1+e^{-x}}$. We will see later that the logit model expressed above has been heavily used in the RLHF step to learn a reward function from preference data.

Perturbed Utility: Stochastic choice can also arise as the solution to expected reward maximization subject to an additive perturbation [9] which may describe limited information processing capacity [10] or aversion to ambiguity [11]. Assuming that the reward function $r(\cdot)$ is known, then the optimal stochastic choice can be modeled as the following utility maximization problem

$$\pi_r^*(\cdot|s) := \arg\max_{\pi \in \Delta^{|A|}} \{\mathbb{E}_{a\sim\pi}[r(s,a)] - \beta \times c(\pi)\}, \tag{4}$$

where $\Delta^{|A|} \subset \mathbb{R}^{|A|}$ is the simplex; $c : \Delta^{|A|} \mapsto \mathbb{R}$ is a convex function; $\beta > 0$ is a constant. For example, when c is the (forward) Kullback-Leibler (KL) divergence with respect to a default policy π^0 defined as

$$D_{\mathrm{KL}}(\pi\|\pi^0) := \sum_{a\in A} \pi(a) \log \frac{\pi(a)}{\pi^0(a)}, \tag{5}$$

then the optimal stochastic choice can be expressed as follows:

$$\pi_r^*(a|s) = \frac{\pi^0(a|s)\exp(\beta^{-1}r(s,a))}{\sum_{\tilde{a}\in A} \pi^0(\tilde{a}|s)\exp(\beta^{-1}r(s,\tilde{a}))}. \tag{6}$$

It should be noted that in order to precisely calculate the optimal policy, one has to assume that all available actions in A are known. Nevertheless, the perturbed utility model described here is instrumental for establishing a number of LLM alignment approaches, assuming that a reward model is available.

B. The reward learning problem

Based upon the observed preferences $\mathcal{D}_{\text{pref}} := \{(s,a) \succ (s,a')\}$ by an agent and the model (3), the estimation problem consists of finding a reward function $r(s,a)$ that *rationalizes* the data, so that observed preferences are *consistent* with a reward model. Specifically, the likelihood of $\mathcal{D}_{\text{pref}}$ can be written as

$$\ell_{\text{pref}}(r) = \mathbb{E}_{\mathcal{D}_{\text{pref}}}\Big[\log P_r\Big((s,a) \succ (s,a')\Big)\Big]. \tag{7}$$

In general, there is no unique reward function that maximizes likelihood $\ell_{\text{pref}}(r)$. However, assuming the reward $r(s,a_0)$ for a reference action $a_0 \in A$ is known, it can be shown that there is unique reward function that maximizes likelihood function $\ell_{\text{pref}}(r)$.

Alternatively, based upon the observed choices $\mathcal{D}_{\text{demo}} := \{(s,a)\}$ and assuming these choices are *consistent* with the model (6). Then the estimation problem consists of finding a reward function $r(s,a)$ that maximizes the likelihood of $\mathcal{D}_{\text{demo}}$ defined as:

$$\ell_{\text{demo}}(r) = \mathbb{E}_{\mathcal{D}_{\text{demo}}}\Big[\log \pi_r^*(a|s)\Big]. \tag{8}$$

As pointed out before, a limitation of this model is the assumption of complete information on the menu of available choices A. When the reward is linearly parametrized, i.e. $r(s,a) = \phi(s,a)^\top\theta$ where $\phi(s,a), \theta \in \mathbb{R}^p$ and $\phi(s,a)$ is a vector of features, problem (8) is the Lagrangian dual of the *maximum entropy* estimation problem (see Theorem 2 in [12]):

$$\begin{aligned} \max_{\pi(\cdot|s)\in\Delta^{|A|}} \quad & \mathbb{E}_{s\sim\mathcal{D}_{\text{demo}}}[\mathcal{H}(\pi(\cdot|s))] \\ \text{s.t.} \quad & \mathbb{E}_{s\sim\mathcal{D}_{\text{demo}}, a\sim\pi(\cdot|s)}[\phi(s,a)] = \mathbb{E}_{(s,a)\sim\mathcal{D}_{\text{demo}}}[\phi(s,a)] \end{aligned}$$

IV. State-of-the-art Algorithms for LLM Alignment

In this section, we provide an overview of the mathematical formulation of the alignment problem, and a number of state-of-the-art algorithms. To facilitate discussion, this section is organized according to the *structure* of the available data.

A. *Learning from demonstrations data*

We begin with the discussion on how to model and learn from *demonstration data* $\mathcal{D}_{\text{demo}} := \{(x,y)\}$. Given the prompt x, the response y is viewed as a *golden answer*. A natural approach to learning from such a dataset is supervised learning, leading to the supervised fine-tuning (SFT) paradigm. To this end, we formulate the following optimization problem over the LLM parameters class Θ:

$$\min_{\theta\in\Theta} \ \ell_{\text{SFT}}(\theta) := -\mathbb{E}_{x\sim\rho, y\sim\pi^{\text{E}}(\cdot|x)}\left[\log \pi\left(y|x;\theta\right)\right]. \tag{9}$$

It is easy to see that the above problem shares the same optimal solutions as

$$\min_{\theta\in\Theta} \ \mathbb{E}_{x\sim\rho}[D_{\text{KL}}(\pi^{\text{E}}\left(\cdot|x\right) \| \pi\left(\cdot|x;\theta\right))], \tag{10}$$

where $D_{KL}(\cdot)$ is the (forward) KL-divergence defined in (5). The latter shows that SFT aims at imitating the demonstration dataset via minimizing the KL divergence.

It is worth noting that the SFT stage described here is closely related to behavioral cloning [13], whose goal is to *mimic* the policy of an expert, we describe the details in the next paragraph.

Protocol 1 SFT (Supervised Fine Tuning) via Behavioral Cloning

```
1: The learner receives: (i) A dataset $\mathcal{D}_{\text{demo}} = \{(x^i, y^i)\}_{i=1}^{N_{\text{demo}}}$, (ii) A policy function parameters class $\Theta$.
2: The learner computes the loss: $\widehat{\ell}_{\text{SFT}}(\theta) := \sum_{i=1}^{N_{\text{demo}}} \left[-\log \pi \left(y^i | x^i; \theta\right)\right]$.
3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta\in\Theta} \widehat{\ell}_{\text{SFT}}(\theta)$.
```

1) Behavioral cloning approaches: Behavioural cloning (see Protocol 1) it is an *offline* problem where the learner cannot collect new data using the current policy, but should learn relying exclusively on the expert dataset. This can be done via a reduction to supervised learning where the actions are the labels to be predicted and the states are the features. Very recently, [14] show that exact minimization of the loss $\widehat{\ell}_{\text{SFT}}$ allows to output a policy which enjoys suboptimality guarantees compared to the expert. Their experiments and theoretical analysis also show that the guarantees and practical performance of the method do not deteriorate for long horizon problems. Unfortunately, exact minimization of the loss $\widehat{\ell}_{\text{SFT}}$ is not possible, and to our knowledge, there are currently no studies investigating how those optimization errors affect performance. It is important to notice that in LLM the environment is simpler than what generally assumed in analyzing and empirically testing BC. Indeed, in language tasks LLMs are tree shaped and deterministic.

2) Self generation approaches: An alternative approach, dubbed SPIN (Self-Play fIne-tuNing), presented in [15] casts the fine-tuning with demonstration problem as a two-player game. On the one hand, the first player tries to generate answers as similar as possible as the observed ones in $\mathcal{D}_{\text{demo}}$. On the other hand, the second player's aims at distinguish artificially generated answers from the answers in the demonstrations dataset $\mathcal{D}_{\text{demo}}$. The algorithm employs alternating updates between the first and second player. In particular, at iteration t, the first player observes the second players parameters θ_t and updates its weights via the following minimization over potentially time-varying parameters classes $\{\mathcal{F}_t\}$ as

$$f_{t+1} = \arg\min_{f\in\mathcal{F}_t} \mathbb{E}_{x,y\sim\mathcal{D}_{\text{demo}}} \mathbb{E}_{y'\sim\pi(\cdot|x;\theta_t)} \left[\ell(f(x,y) - f(x,y'))\right], \tag{11}$$

where $\ell(\cdot)$ is a monotonically decreasing, non negative, smooth and convex function. A common choice in practice satisfying the above properties is the logistic loss function $\ell(x) = \log\big(1+\exp(-x)\big)$. All these properties are imposed to make the optimization problem easier to solve via first order methods. At this point, the second player can update its weights exploiting knowledge of f_{t+1}. This is done by solving the following problem: $\theta_{t+1} = \arg\min_{\theta\in\Theta} -\mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x;\theta_t)} \left[f_{t+1}(x,y) + \beta D_{\text{KL}}(\pi(\cdot|x;\theta)||\pi(\cdot|x;\theta_t))\right]$, which is an application of Equation (4) for $c(\pi) = -D_{\text{KL}}(\pi(\cdot|x)||\pi(\cdot|x;\theta_t))$. The analytical solution of the above problem is surprisingly easy to obtain (analogously to (6)): $\pi(y|x;\theta_{t+1}) \propto \pi(y|x;\theta_t)\exp\left(\frac{f_{t+1}(x,y)}{\beta}\right)$, if the class Θ is expressive enough compared to the class $\mathcal{F}_{t+1}$. To ensure that this is the case, in [15] the

```
Protocol 2 SFT (Supervised Fine Tuning) via SPIN (Self-Play fIne tuNing)
1: The learner receives: (i) A dataset $\mathcal{D}_{\text{demo}} = \{(x^i, y^i)\}_{i=1}^{N_{\text{demo}}}$, (ii) A policy function parameters class $\Theta$.
2: The learner samples $y'^{,i} \sim \pi(\cdot|x^i;\theta_t)$ and computes the loss:
   $$\widehat{\ell}_{\text{SPIN}}(\theta) := \sum_{i=1}^{N_{\text{demo}}} \left[\ell\left(\beta \log\left(\frac{\pi(y^i|x^i;\theta)}{\pi(y^i|x^i;\theta_t)}\right) - \beta \log\left(\frac{\pi(y'^{,i}|x^i;\theta)}{\pi(y'^{,i}|x^i;\theta_t)}\right)\right)\right].$$
3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta\in\Theta} \widehat{\ell}_{\text{SPIN}}(\theta)$.
```

class $\mathcal{F}_{t+1}$ is chosen to be coupled with Θ as follows, $\mathcal{F}_{t+1} = \left\{\beta \log\left(\frac{\pi(y|x;\theta_{t+1})}{\pi(y|x;\theta_t)}\right)\right\}$. Another benefit of this choice is that the two updates can now be expressed as single update over the weight class Θ. Indeed, (11) is at this point equivalent to

$$\theta_{t+1} = \arg\min_{\theta\in\Theta} \ell_{\text{SPIN}}(\theta) = \mathbb{E}_{x,y\sim\mathcal{D}_{\text{demo}}}\mathbb{E}_{y'\sim\pi(\cdot|x;\theta_t)}\left[\ell\left(\beta \log\left(\frac{\pi(y|x;\theta)}{\pi(y|x;\theta_t)}\right) - \beta \log\left(\frac{\pi(y'|x;\theta)}{\pi(y'|x;\theta_t)}\right)\right)\right]. \tag{12}$$

The practical algorithm can be obtained via a stochastic estimator of this loss constructed with the dataset $\mathcal{D}_{\text{demo}}$, as described in Protocol 2. We note that SPIN shares similarity with Direct Preference Optimization (DPO), to be described in Section IV-C, however there are also important differences that we will highlight in Section V.

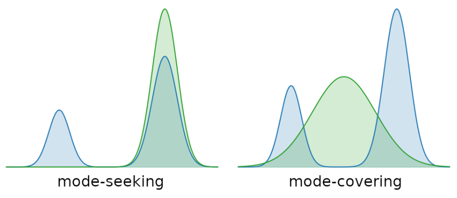


Fig. 4: True distribution in blue and estimated distribution in green. When the estimated distribution has a smaller support than the true one, as in the left plot, we say that it has a mode seeking. Vice versa, the estimated distribution has a mode covering behavior if its support is larger, as in the right plot. Image from https://sander.ai/2020/03/24/audio-generation.html.

3) Information theoretic approaches: Recent works [16], [17] considered matching general information theoretic divergences between answers distribution.

As mentioned, minimizing the forward KL divergence between demonstrated and learner answer distribution is equivalent to the SFT objective in (10). As commonly noticed, the forward KL minimization outputs a mode covering distribution. That is, the learned distribution has support which contains the support of the demonstrations distribution. However, a mode seeking behavior can be more desirable if one wants to perfectly imitate some aspects of the demonstrations while discarding others. We refer the reader to Figure 4 for a visual illustration. To this end, [16] suggested to minimize the reversed KL divergence. This method however requires additional prompts and answers generated by the learner policy. As a last example of divergence minimization, [18] considers χ^2 divergence minimization.

Protocol 3 RLHF (Reinforcement Learning from Human Feedback)

```
1: The learner receives: (i) A preference dataset $\mathcal{D}_{\text{pref}} = \{(x^i, y_w^i, y_\ell^i)\}_{i=1}^{N_{\text{pre}}}$, (ii) A prompt dataset $\mathcal{D}_{\text{prompts}} = \{x^i\}_{i=1}^{N_{\text{prompts}}}$, (iii) A reward function parameters class $\Phi$, (iv) A policy function parameter class $\Theta$.
2: The learner estimates the reward as $r(\cdot, \cdot; \phi^\star)$ with
       $$\phi^\star = \arg\min_{\phi \in \Phi} \ \widehat{\ell}_{\text{RM}}(\phi) := -\sum_{i=1}^{N_{\text{pre}}} \log \Big(\sigma\big(r(x^i, y_w^i; \phi) - r(x^i, y_l^i; \phi)\big)\Big).$$
3: The learner finds $\theta^\star \approxeq \arg\min_{\theta \in \Theta} \ell_{\text{RL}}(\theta)$, using PPO or REINFORCE.
4: The learner outputs $\pi_{\theta^\star}$.
```

Protocol 4 PPO (Proximal Policy Optimization) [19]

```
1: Receive policy class parameters $\Theta$, and reward model parameters $\phi^\star$. Initialize $\theta_1$.
2: for $t = 1, \dots, T$ do
3:     Sample $y^i \sim \pi(\cdot|x^i; \theta_t)$ and $x^i \in \mathcal{D}_{\text{prompts}}$.
4:     The learner computes the stochastic loss.
           $$\widehat{\ell}_{\text{RLHF}}(\theta_t) := -\frac{1}{|\mathcal{D}_{\text{prompts}}|} \sum_{i \in [N_{\text{prompts}}]} \left[r(x^i, y^i; \phi^\star) + \beta \log \pi(y^i|x^i; \theta_t) - \beta \log \pi_{\text{ref}}(y^i|x^i)\right],$$
5:     The learner computes the policy ratio $\delta(\theta; \theta_t) = \frac{1}{|\mathcal{D}_{\text{prompts}}|} \sum_{i \in [N_{\text{prompts}}]} \frac{\pi(y^i|x^i;\theta)}{\pi(y^i|x^i;\theta_t)}$.
6:     The learner computes $\ell_{\text{PPO}}(\theta) = -\min\Big(\delta(\theta; \theta_t)\widehat{\ell}_{\text{RLHF}}(\theta_t), \text{Clip}(\delta(\theta; \theta_t), 1+\epsilon, 1-\epsilon)\widehat{\ell}_{\text{RLHF}}(\theta_t)\Big)$
7:     $\theta_{t+1} \approxeq \arg\min_{\theta \in \Theta} \ell_{\text{PPO}}(\theta)$ computed via SGD, Adam or other optimizers.
8: end for
9: The learner outputs $\pi_{\theta^\star}$.
```

B. Learning from preferences and prompts

Learning from preferences and prompts is usually performed in two steps: (i) reward learning using the preference dataset (ii) based on the reward model learned in the first step, further fine-tune the LLM by conducting reinforcement learning using the prompt datasets. Note that these two steps are closely related to the two types of stochastic choice models discussed in Sec. III-A.

a) Reward learning: To find an appropriate reward model, RLHF (see e.g., [21]) leverages a set of *preference dataset* $\mathcal{D}_{\text{pref}} := \{(x, y_w, y_l)\}$, where each data contains a pair of output y_w, y_l. The output y_w is preferred over y_l by human labeler (denoted as $y_w \succ y_l$). Leveraging the the Bradley-Terry model (BT) discussed in (3) and the likelihood of the preference data (7), one could formulate the following problem to find the reward model (parameterized by ϕ):

$$\phi^\star = \arg\min_{\phi \in \Phi} \ \ell_{\text{RM}}(\phi) := -\mathbb{E}_{x \sim \rho, (y_w \succ y_l) \sim \pi^P(\cdot|x)} \Big[\log \Big(\sigma\big(r(x, y_w; \phi) - r(x, y_l; \phi)\big)\Big)\Big]. \tag{13}$$

Protocol 5 REINFORCE [20]

```
1: Receive policy class parameters $\Theta$ and reward model parameters $\phi^\star$. Initialize $\theta_1$, step size $\eta$
2: for $t = 1, \dots, T$ do
3:     Sample $y^i \sim \pi(\cdot|x^i;\theta_t)$ and $x^i \in \mathcal{D}_{\text{prompts}}$.
4:     The learner computes the stochastic loss.
           $\widehat{\ell}_{\text{RLHF}}(\theta_t) := -\frac{1}{|\mathcal{D}_{\text{prompts}}|}\sum_{i\in[N_{\text{prompts}}]}\left[r(x^i,y^i;\phi^\star) + \beta\log\pi(y^i|x^i;\theta_t) - \beta\log\pi_{\text{ref}}(y^i|x^i)\right],$
5:     The learner performs the update $\theta_{t+1} = \Pi_\Theta\left[\theta_t - \eta\widehat{\ell}_{\text{RLHF}}(\theta_t)\sum_{i\in[N_{\text{prompts}}]}\nabla\log\pi(y^i|x^i;\theta_t)\right]$.
6: end for
7: The learner outputs $\pi_{\theta^\star}$.
```

Protocol 3 describes the procedure where $r(\cdot;\phi)$ is learned via minimization of the empirical loss $\widehat{\ell}_{\text{RM}}$.

b) Policy Learning: Once a reward model $r(x, y; \phi^\star)$ is learned, the LLM can be fine tuned by leveraging the perturbed utility model (4), which results in the following problem:

$$\min_\theta\ \ell_{\text{RL}}(\theta) := -\mathbb{E}_{x\sim\rho,y\sim\pi(\cdot|x;\theta)}\left[r(x,y;\phi)\right] + \beta\mathbb{E}_{x\sim\rho}[D_{\text{KL}}(\pi\left(\cdot|x;\theta\right)\|\pi_{\text{ref}}\left(\cdot|x\right))], \tag{14}$$

where π_{ref} is a fixed reference model such as the SFT-ed model, and β is the temperature coefficient of the KL regularizer. Note that both the expected reward and the KL in (14) are *not* computable given the large cardinality of $\mathcal{X}$ and $\mathcal{Y}$, so (14) is usually solved approximately by minimizing an empirical loss, using policy optimization techniques such as REINFORCE [20] or PPO [19]; see Protocol 3 .

Pros and Cons: PPO or REINFORCE? PPO was proposed as an improvement over the REINFORCE algorithm in environment where the gradient estimates generated by REINFORCE suffer from high variance. As a result, PPO enforces conservative policy updates to counterbalance the high variance effect. However, as argued in [22], for the LLM fine-tuning task the gradient variance is not as high to justify the PPO conservative mechanism. Since both options are used across the community, we report both algorithms in Protocols 4 and 5 respectively. A disadvantage of PPO is that it requires a critic network to estimate the advantage function when implemented in multi-stage problems, e.g., when the reward is assigned at the token level rather than to the entire answer y. REINFORCE does not require training a critic, therefore it is more memory and computationally efficient than PPO. Recently, a new policy optimization method (GRPO) has been introduced and has been proven successful in reasoning applications (see Section IV-E).

Pros and Cons: Learning from demonstrations vs learning from preferences. It is widely observed that the models learned *only* using SFT stage do not perform as well as those that are trained via episodically learning the policy (14) and learning the reward (13) on top of the SFT models [23]. This is because the reward model and the RL algorithms together guide the LLMs to explore high-quality responses that can go beyond those defined by the SFT data, thus improving their generalization capabilities. Moreover, obtaining the SFT dataset is typically known to be more expensive than obtaining preference data, therefore standard alignment process leverages *both* the SFT and the preference data.

C. *Learning from preferences without online generations: Direct Preference Optimization*

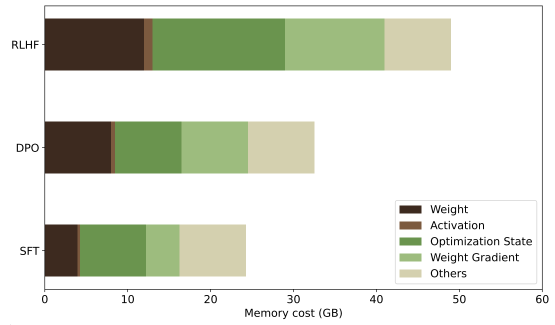


Fig. 5: Estimated memory consumption of running the `Pythia-1B` model on `TL;DR` dataset with batch-size 2 on a single device

Direct Preference Optimization (DPO) [24] bypasses the need to perform the RL, by noticing that minimizing $\ell_{\mathrm{RL}}(\cdot)$ admits an analytical solution if we optimize over the whole Marvov stationary policy space denoted by Π. In fact, notice that from our discussion on stochastic choice model (6), for a fixed $r(\cdot,\cdot;\phi)$, the solution to the policy optimization problem $\pi^\star(\cdot,x;\phi) = \arg\min_{\pi\in\Pi} -\mathbb{E}_{y\sim\pi(\cdot|x)}\left[r(x,y;\phi)\right] + \beta D_{\mathrm{KL}}(\pi(\cdot|x)||\pi_{\mathrm{ref}}(\cdot|x))$ is given below

$$\pi^\star_\phi(y|x) = \frac{\pi_{\mathrm{ref}}(y|x)\exp(\beta^{-1}r(x,y;\phi))}{Z(x,\phi,\beta)}, \tag{15}$$

where we defined $Z(x,\phi,\beta) := \sum_{y'}\pi_{\mathrm{ref}}(y'|x)\exp(\beta^{-1}r(x,y';\phi))$. This implies that for all y, it holds that $\log(\pi^\star_\phi(y|x)) = \log(\pi_{\mathrm{ref}}(y|x)) + \beta^{-1}r(x,y;\phi) - \log(Z(x,\phi,\beta))$. Such a quantity cannot be computed in closed form because computing $Z(x,\phi,\beta)$ is intractable. However, for two possible answers y, y' to the *same* question x, it holds that $\log(\pi^\star_\phi(y|x)) - \log(\pi^\star_\phi(y'|x)) = \log(\pi_{\mathrm{ref}}(y|x)) - \log(\pi_{\mathrm{ref}}(y'|x)) + \beta^{-1}r(x,y;\phi) - \beta^{-1}r(x,y';\phi)$. Therefore, the difference of reward functions for different answers y, y' can be computed efficiently as the normalization constant $Z(x,\phi,\beta)$ does not appear, i.e.

$$r(x,y;\phi) - r(x,y';\phi) = \beta\log\left(\frac{\pi^\star_\phi(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right) - \beta\log\left(\frac{\pi^\star_\phi(y'|x)}{\pi_{\mathrm{ref}}(y'|x)}\right). \tag{16}$$

It follows that we can plug in the above analytical solution into the loss ℓ_{RM} (13) to obtain the following bi-level optimization problem that finds the optimal reward parameterization:

$$\min_{\phi\in\Phi}\ \ell(\phi) := -\mathbb{E}_{x\sim\rho,(y_l\prec y_w)\sim\pi^P(\cdot|x)}\left[\log\left(\sigma\left(\beta\log\left(\frac{\pi^\star_\phi(y_w|x)}{\pi_{\mathrm{ref}}(y_w|x)}\right) - \beta\log\left(\frac{\pi^\star_\phi(y_l|x)}{\pi_{\mathrm{ref}}(y_l|x)}\right)\right)\right)\right] \tag{17a}$$

$$\text{s.t.}\quad \pi^\star_\phi := \arg\max_{\pi\in\Pi}\ \mathbb{E}_{x\sim\rho,y\sim\pi(\cdot|x)}\left[r(x,y;\phi)\right] - \beta\mathbb{E}_{x\sim\rho}[D_{\mathrm{KL}}(\pi(\cdot|x)\,\|\pi_{\mathrm{ref}}(\cdot|x))], \tag{17b}$$

where $\pi^\star_\phi$ is an optimal policy defined in (15) under a certain reward model parameterized by ϕ. If one faithfully follows the above approach, then one would first compute the optimal reward $r(\cdot;\phi^*)$, then

Protocol 6 DPO (Direct Preference Optimization)

```
1: The learner receives as input:
     • A preference dataset $\mathcal{D}_{\text{pref}} = \{(x^i, y_w^i, y_\ell^i)\}_{i=1}^N$.
     • A policy function parameters class $\Theta$.
2: The learner computes the stochastic loss.
     $\widehat{\ell}_{\text{DPO}}(\theta) := -\frac{1}{|\mathcal{D}_{\text{pref}}|} \sum_{x,y_w,y_l \in \mathcal{D}_{\text{pref}}} \left[\log\left(\sigma\left(\beta \log\left(\frac{\pi(y_w|x;\theta)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta \log\left(\frac{\pi^\star(y_l|x;\theta)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right)\right]$
3: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta\in\Theta} \widehat{\ell}_{\text{DPO}}(\theta)$.
```

use it to identify the optimal policy using (15). This is still too complicated, so [24] proposes to directly parameterize the policy and find the parameters $\theta^\star$ that solve the following problem:

$$\arg\min_{\theta\in\Theta} \ \ell_{\text{DPO}}(\theta) := -\mathbb{E}_{x\sim\rho,(y_l \prec y_w)\sim\pi^P(\cdot|x)} \left[\log\left(\sigma\left(\beta \log\left(\frac{\pi(y_w|x;\theta)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta \log\left(\frac{\pi(y_l|x;\theta)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right)\right]. \tag{18}$$

Notice that, as shown in Protocol 6, an empirical estimate of this loss can be computed without the need of the dataset $\mathcal{D}_{\text{prompts}}$ and online generations. A first clear advantage of DPO over RLHF is that only the policy variable is kept in memory. The reward network does not need to be stored, therefore, as shown in the Fig. 5, DPO is more memory efficient than RLHF. Further, no self-generation such as line 3 of Protocol 4 and 5 is needed, so DPO is much easier to implement as compared to RL based algorithms.

Action	y_1	y_2	y_3
π_{ref}	0.5	0.5	0
D_{pref}	$\{(y_w = y_1, y_l = y_2)\}$		
π_{DPO}	α	0.0	$1-\alpha$
π_{RLHF}	1	0	0

TABLE I: The policy π_{DPO} minimizes ℓ_{DPO} for any $\alpha \in [0,1)$. Hence, for $\alpha > 0$, it produces a policy with support which is not a subset of π_{ref}. The same can not happen for π_{RLHF} that minimizes ℓ_{RLHF}.

However, dropping the analytical form of $\pi^\star_\phi(y|x)$ in the last step of the DPO derivation creates practical differences between the performance of DPO and RLHF. Therefore, DPO and RLHF are not equivalent, in particular, since we dropped a constraint, the solution set for DPO is a superset of the solution set for RLHF. Recent studies suggest that RL-based fine-tuning has greater potential to enhance the performance of LLMs compared to the DPO method [25]

From the formulation of DPO in Eq. (18), we see that DPO only fits the given preference data and does not leverage online generations from its own model (see line 3 of Protocol 4 and 5). Due to the lack of the optimal policy constraint, the model trained by the DPO method can suffer from the limited generalization capability and can encounter the distribution shift issue when the coverage of the preference dataset is limited.

Furthermore, when there exists distribution shift between the model outputs and the preference dataset, the performance of DPO can be significantly affected. In contrast, in the RLHF training pipeline, the policy model is trained by RL and has leveraged online generations from the model itself, which can alleviate the distribution shift issue between model outputs and the preference dataset through leveraging the generalization power from the explicit estimated reward model. Similar empirical observations for the potential drawbacks of DPO compared with RLHF / PPO has also been observed in recent study [25]. Hence, to unlock the full potential of LLMs, it appears that RL is a necessary step in the post-training pipeline.

	PPO Win	Tie	DPO Win
PPO V.S. DPO	42	28	30

TABLE II: On HH-RLHF, GPT-4 is used to decide the winner between PPO and DPO models' output. Taken from [25].

The toy example in [25] that we report in Table I shows that the solution set of DPO is a strict superset of the solution set of RLHF. In particular, there are minimizers of the DPO loss assign positive probability to answers for which no information is available (y_3 in this example). This example shows that DPO should not be applied when the preference dataset has poor coverage of the answers set $\mathcal{Y}$. For more practical problems, it is also observed that DPO can be less performant in terms of winrate against the model fine tuned via RLHF; see Table II.

D. *Learning from preference and prompts without the BT assumption.*

Regardless of using RLHF and the DPO approach to perform alignment, one key step is to use the BT model (13) to model human preference. The BT assumption implies transitivity. This is restrictive because the preference dataset collected from different humans might not be transitive even if each human follows a transitive model in generating the preference. As an example, let us consider 3 humans h_1, h_2, h_3 and 3 possible answers y_1, y_2, y_3. Let us denote P_h the preference model of human h. Then, we have that for the humans h_1, h_2, h_3 follows these preference models.

$$\begin{array}{lll}
\mathcal{P}_{h_1}(y_1 \succ y_2) = 1 & \mathcal{P}_{h_1}(y_2 \succ y_3) = 0 & \mathcal{P}_{h_1}(y_3 \succ y_1) = 1 \\
\mathcal{P}_{h_2}(y_1 \succ y_2) = 0 & \mathcal{P}_{h_2}(y_2 \succ y_3) = 1 & \mathcal{P}_{h_2}(y_3 \succ y_1) = 1 \\
\mathcal{P}_{h_3}(y_1 \succ y_2) = 1 & \mathcal{P}_{h_3}(y_2 \succ y_3) = 1 & \mathcal{P}_{h_3}(y_3 \succ y_1) = 0.
\end{array}$$

Notice, that each of these models is transitive. However, the average model defined as $\mathcal{P}(y \succ y') = \frac{1}{3}\sum_{h\in\{h_1,h_2,h_3\}} \mathcal{P}_h(y \succ y')$ satisfies $\mathcal{P}(y_1 \succ y_2) = \mathcal{P}(y_2 \succ y_3) = \mathcal{P}(y_3 \succ y_1) = 2/3$. That is, the average model is non transitive and can not be modeled by the BT assumption. Therefore, the BT assumption is data wasteful. In this example, one should consider preferences only from a single human in order to

Protocol 7 NLHF (Nash Learning from Human Feedback)

```
1: The learner receives as input: (i) A preference dataset $\mathcal{D}_{\text{pref}} = \{(x^i, y^i_w, y^i_\ell)\}_{i=1}^N$, (ii) a prompt dataset $\mathcal{D}_{\text{prompts}} = \{x^i\}_{i=1}^{N_{\text{prompts}}}$, (iii) a preference model function parameters class $\Phi$ and (iv) a policy function parameters class $\Theta$.
2: The learner estimates the preference model as $\mathcal{P}(\cdot \succ \cdot|\cdot;\phi^\star)$ with
   $$\phi^\star = \arg\min_{\phi\in\Phi} \widehat{\ell}_{\text{PM}}(\phi) := -\sum_{i=1}^N \log\left(\mathcal{P}(y^i_w \succ y^i_l|x^i;\phi)\right).$$
3: Sample $y^i \sim \pi(\cdot|x^i;\theta)$ and $y'^{,i} \sim \pi(\cdot|x^i;\theta')$ for all $i \in [N]$.
4: The learner computes the stochastic objective
   $$\widehat{\ell}_{\text{NLHF}}(\theta,\theta') := -\frac{1}{|\mathcal{D}_{\text{prompts}}|}\sum_{i=1}^N \mathcal{P}(y^i \succ y'^{,i}|x^i;\phi^\star) + \beta D_{\text{KL}}(\pi(\cdot|x^i;\theta)\,\|\pi_{\text{ref}}(\cdot|x^i)) - \beta D_{\text{KL}}(\pi(\cdot|x^i;\theta')\,\|\pi_{\text{ref}}(\cdot|x^i)),$$
5: The learner outputs $\pi_{\theta^\star}$ with $\theta^\star = \arg\min_{\theta\in\Theta}\max_{\theta'\in\Theta}\widehat{\ell}_{\text{NLHF}}(\theta,\theta')$.
```

make the BT assumption valid. Not enforcing the BT assumption, allows to use more data, i.e. preferences from all the three humans. Given this motivation, recent works [26] investigated learning from preferences directly. The training happens in two stages as in RLHF: the first step uses maximum likelihood to learn a preference model rather than a reward function, and a second step identifies a Nash equilibrium rather than an optimal policy.

a) Preference learning: First, we learn via maximum likelihood, a preference model $\mathcal{P} : \mathcal{X}\times\mathcal{Y}\times\mathcal{Y} \to [0,1]$ such that $\mathcal{P}(y \succ y'|x)$ is the probability that certain oracle that produced the preferences in the dataset $\mathcal{D}_{\text{pref}}$ prefers answer y over y' following the prompt x. Mirroring the procedure in RLHF, let us consider a preference model class with parameters class Φ, i.e. $\{\mathcal{P}(\cdot \succ \cdot|\cdot;\phi)\}_{\phi\in\Phi}$, consider the following minimization problem (where PM stands for preference modelling):
$\phi^\star = \arg\min_{\phi\in\Phi}\ \ell_{\text{PM}}(\phi) := -\mathbb{E}_{x\sim\rho,(y_l\prec y_w)\sim\pi^P(\cdot|x)}\left[\log\left(\mathcal{P}(y_w \succ y_l|x;\phi)\right)\right]$. Protocol 7 shows how to do this practically, minimizing an empirical unbiased estimate for ℓ_{PM}.

b) Nash equilibrium learning: After learning the preference model, [26] proposes to find the unique Nash equilibrium of the objective $\widehat{\ell}_{\text{NLHF}}$ in Protocol 7. Due to the min max structure of this problem, the optimization side is more delicate and require careful conservative updates to ensure last iterate convergence. However, this additional complexity on the optimization side comes at the benefit of dropping the BT assumption and therefore accommodating a larger number of preference data.

Experiments in [26, Table 1] shows that the NLHF method dubbed MD1 achieves win rate of 0.598 against the same model aligned via RLHF on a text summarization task. The winner model is decided

Protocol 8 GRPO (Group Relative Policy Optimization) [27]

```
1: Receive policy class parameters $\Theta$, and reward model parameters $\phi^\star$. Initialize $\theta_1$.
2: for $t = 1, \ldots, T$ do
3:     Sample $G$ outputs $\{y^{i,j}\}_{j=1}^{G} \sim \pi(\cdot|x^i; \theta_t)$ and $x^i \in \mathcal{D}_{\text{prompts}}$.
4:     The learner computes rewards $\{r_j\}_{j=1,\ldots,G}$ for each output $y^{i,j}$.
5:     The learner computes the estimated advantage function $\hat{A}_j = (r_j - \text{mean}(\{r_j\}))/\text{std}(\{r_j\})$
6:     The learner computes
       $$\ell_{\text{GRPO}}(\theta) = \frac{1}{|\mathcal{D}_{\text{prompts}}|} \sum_{i \in [N_{\text{prompts}}]} \frac{1}{G} \sum_{j=1}^{j} \left[ \frac{\pi(y^{i,j}|x^i;\theta)}{\pi(y^{i,j}|x^i;\theta_t)} \hat{A}_j - \beta \Big( \frac{\pi(y^{i,j}|x^i;\theta)}{\pi(y^{i,j}|x^i;\theta_t)} - \log \frac{\pi(y^{i,j}|x^i;\theta)}{\pi(y^{i,j}|x^i;\theta_t)} - 1 \Big) \right]$$
7:     $\theta_{t+1} \approxeq \arg\min_{\theta\in\Theta} \ell_{\text{GRPO}}(\theta)$ computed via SGD, Adam or other optimizers.
8: end for
9: The learner outputs $\pi_{\theta^\star}$.
```

quering a large LLM. The drawback of this method is that it requires solving a minmax optimization problem rather than the simpler minimization problem required in RLHF.

E. Learning from data verifiers via Rule-Based Reinforcement Learning

Before closing this section, we would like to mention a recently developed alignment strategy, which is fundamental for training powerful reasoning models such as DeepSeek-R1 [28]. The general idea is simple: To teach the model to solve many challenging math, STEM or coding related problems, a reward model is no longer needed. Rather, one can simply use a *solution verifier*, which simply verifies whether the solution is correct or not. In [28], the authors proposed to use LeetCode tests case as verifier for programming questions, while creating a dataset of mathematical problems from OpenWebMath for math verifiers. In particular, in the reproduction of the Deepseek R1, [29] adopted the rule-based reward system that comprises two types of rewards: Format Reward and Answer Reward. Here, we show the detailed reward designed in [29] as below:

$$r_{\text{format}} = \begin{cases} 1, & \text{if format is correct} \\ -1, & \text{if format is incorrect} \end{cases}, \quad r_{\text{answer}} = \begin{cases} 2, & \text{if the answer fully matches the ground truth} \\ -1.5, & \text{if the answer partially mismatches the ground truth} \\ -2, & \text{if the answer cannot be parsed or is missing} \end{cases}$$

where r_{format} is the format reward and r_{answer} is the answer reward.

However in [28], it is noted that the model which optimizes purely the rule based reward, i.e. DeepSeek-R1-Zero leads to poor language skills suffering for example from language mixing. To avoid this fact, [28] uses multi-stage training with learning with verifiable rewards at the initial stage and SFT/preference based learning used at the final stages to improve the language skills of the model. In the initial stage, the rule based reward is optimized via GRPO in Algorithm 8.

V. A Unified Perspective for LLM Alignment

So far we have discussed a number of state-of-the-art approaches for LLM alignment. Generally speaking, they can be divided into two categories: alignment using the demonstration data, and alignment using the preference data. One might wonder why algorithms belonging to these two categories are so different (e.g., supervised learning vs. reward learning + RL). After all, both families of algorithms are trying to learn from some forms of human preference. In this section, we provide a perspective derived from the stochastic choice model discussed in Section III, which serves to unify different approaches of alignment.

Recall that in Section III, we have introduced two estimation problems (7) and (8). Although they use different data sets (i.e. the former uses the preference dataset $\mathcal{D}_{\text{pref}}$, while the latter uses the demonstration dataset $\mathcal{D}_{\text{demo}}$), they share a few common properties: (1) both learn reward functions, and (2) both maximize some form of the likelihood function. These observations lead to the following questions:

- (Q1) In the alignment algorithms discussed in the previous section, why do we only learn reward from the preference dataset, but not from the demonstration dataset?
- (Q2) Since both problems maximize the likelihood functions, why do we need to separate the alignment process into different stages (i.e., SFT and RLHF)?

In the following, we will first address question (Q1) by formulating a new problem that learns *both* the reward and the policy from $\mathcal{D}_{\text{demon}}$. We will show that this formulation is closely related to a class of RL problem called inverse RL (IRL), and we will derive efficient solution methods from the IRL literature. Somewhat unexpectedly, it turns out that a number of algorithms discussed in the previous section, such as SFT and SPIN, can be viewed as special cases of the proposed approach.

Furthermore, leveraging the development in (Q1), we address (Q2) by introducing a framework that unifies the two stages of alignment problem into a *single* stage. The new framework, named Alignment with Integrated Human Feedback (AIHF), jointly learns a reward and a policy by maximizing the likelihood of *both* $\mathcal{D}_{\text{demon}}$ and $\mathcal{D}_{\text{pref}}$. We will demonstrate the generality of AIHF by discussing its connection with various alignment algorithms introduced in the previous section.

A. Jointly learning policy and reward from the demonstrations data

To jointly learn a reward and a policy from the demonstration dataset $\mathcal{D}_{\text{demon}}$, let us leverage the estimation problem (8) derived from the stochastic choice theory. Given the demonstration (x, y) sampled from an expert policy π^{E}, we can directly translate (8) with the optimal reward defined in (4), into the LLM alignment setting and obtain the following problem [30]:

$$\max_{\phi} \ell_{\mathrm{IRL}}(\phi) := \mathbb{E}_{x\sim\rho, y\sim\pi^{\mathrm{E}}(\cdot|x)}\left[\log \pi^{\star}_{r_{\phi}}(y|x)\right] \tag{19a}$$

$$\pi^{\star}_{r_{\phi}} := \arg\max_{\pi} \quad \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x)}\left[r(x,y;\phi)\right] - \beta\mathbb{E}_{x\sim\rho}[D_{\mathrm{KL}}(\pi(\cdot|x)\,\|\pi_{\mathrm{ref}}(\cdot|x))]. \tag{19b}$$

Compared with the SFT formulation in (9), we see that (19) shares the same objective but has an additional constraint which ensures that the policy has to be an optimal policy under a certain reward model.

Problem (19) falls under the class of inverse reinforcement learning (IRL) problems, where the objective is to jointly learn an optimal policy and the underlying reward model, assuming access to expert policy samples. Similarly, as noted in Sec. IV-A1, the conventional SFT formulation (9) is a form of behavioral cloning that aims to replicate an expert's policy. However, behavioral cloning generally exhibits poorer generalization than IRL. Indeed, [31] shows that behavioral cloning is a suboptimal approach to learning from demonstrations, leading to regret bounds that scale quadratically with the problem horizon. This arises due to distribution shift – the learned policy may encounter states not covered in the demonstration data, causing compounding errors that increasingly deviate from the expert trajectory over time.

In contrast, IRL mitigates distribution shift and achieves better generalization by explicitly modeling the expert's reward function, rather than simply mimicking actions. By recovering an estimate of the reward structure that drives expert behavior, IRL enables the agent to make goal-directed decisions even in unseen states. This contrasts with behavioral cloning, which lacks an underlying objective thus often struggles in novel situations. Additionally, optimizing policies with a learned reward function allows for adaptive behavior, where the agent can deviate from expert demonstrations in a way that still aligns with the inferred intent. As a result, IRL-based policies are generally more robust in complex environments where state distributions differ from training data [32]. Recognizing the limitations of standard SFT methods, particularly regarding distribution shift, researchers have proposed a range of IRL-based techniques [33]–[36] to more effectively leverage demonstration data.

Unfortunately, the IRL problem (19) appears to be much involved compared to the SFT problem (9), as it involves *two* levels of optimization problem stacked in hierarchy. It turns out that there is a way to simplify it. In [37], [38], the authors show that when a finite demonstration dataset $\mathcal{D}_{\mathrm{demo}} := \{(x,y)\}$ is available, one can consider the following surrogate objective:

$$\max_{\phi}\min_{\theta} \ \widetilde{L}(\phi,\theta;\mathcal{D}_{\mathrm{demo}}) := \mathbb{E}_{(x,y)\sim\mathcal{D}, y'\sim\pi(\cdot|x;\theta)}\Big[r(x,y;\phi) - r(x,y';\phi) + \beta D_{\mathrm{KL}}\Big(\pi(\cdot|x;\theta)\|\pi_{\mathrm{ref}}(\cdot|x)\Big)\Big]. \tag{20}$$

As a remark, one can plug the closed-form expression of the optimal policy $\pi^*_{r_\phi}$ into the maximum likelihood objective (19a), then one can see that the surrogate objective (20) is a finite-sample version of the IRL problem (19). Further, as the number of demonstration data becomes large, problem (20) becomes asymptotically close to problem (19), in the sense that their optimal solutions become asymptotically close to each other (with provable bounds) [37]. The surrogate objective in equation (20) offers an alternative interpretation of the IRL problem formulated in (19) from the adversarial training perspective. Specifically, the reward estimator is optimized to distinguish between expert demonstrations and self-generated trajectories (the max problem), while the policy is optimized to generate trajectories which can best mimic the expert demonstrations and the self-generated trajectories (the min problem). Therefore, it is reasonable to develop practical algorithms based on (20), which is apparently much easier to optimize. As suggested in [38], one can use the following alternating algorithm to optimize the reward and the policy:

$$\phi_{t+1} := \arg\min_{\phi} \; -\mathbb{E}_{(x,y)\sim\mathcal{D}, y'\sim\pi(\cdot|x;\theta_t)}\Big[\ell\Big(r(x,y;\phi) - r(x,y';\phi)\Big)\Big]. \tag{21a}$$

$$\theta_{t+1} := \arg\max_{\theta} \;\; \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x;\theta)}\left[r(x,y;\phi_{t+1})\right] - \beta\mathbb{E}_{x\sim\rho}[D_{\mathrm{KL}}(\pi\left(\cdot|x;\theta\right)\|\pi_{\mathrm{ref}}\left(\cdot|x\right))]. \tag{21b}$$

Note that if we directly apply the above gradient descent/ascent algorithm to (20), then $\ell(\cdot)$ in (21a) should be a linear function $\ell(x) = x$. Considering that the use of a linear loss function results in an unbounded objective value, one can typically choose some easy-to-optimize loss functions such as log sigmoid function $\ell(z) := \log\sigma(z)$. Next, let us make some remarks about the algorithm (21).

Connection to the RLHF. We observe that (21) is an iterative update scheme which *integrates* the reward and policy updates. This closely resembles the RLHF pipeline, where both the reward and policy are updated. There are two major differences. First, in RLHF the reward and the policy are updated in two *separate* stages, while here they are updated *iteratively*. Second, since only demonstration data $\mathcal{D}_{\text{demon}}$ are available, when learning a reward model in (21a), the demonstration is always treated as the preferred data, while the nonpreferred data are the model-generated responses. This is reasonable, under the assumption that expert data are of high quality. In practice, RLHF-based and IRL-based methods are not competitive but complementary. It's common to leverage both demonstration and preference datasets jointly in the training pipeline to achieve better performance [39], [40]. In [41], [42], the connections between IRL and RLHF have been discussed. In general, it can be shown that RLHF could be a special form of IRL. In (15), given the reward parameter ϕ, the optimal policy has the closed-form $\pi^\star_{r_\phi}(y|x) = \frac{\pi_{\text{ref}}(y|x)\exp(\beta^{-1}r(x,y;\phi))}{Z(x,\phi,\beta)}$. With this result, we can now obtain a model for the likelihood that sequence y_w is preferred over y_l. By the *independence of irrelevant alternatives* property [43] of the optimal choice μ_ϕ, when the set of feasible choices is reduced to just the the two-tuple $\{y_l, y_w\}$, the likelihood that sequence y_w is preferred over

y_l is given by $\mathbb{P}_{\pi^*_{r_\phi}}(y_w \succ y_l|x) := \frac{\pi^*_{r_\phi}(y_w|x)}{\pi^*_{r_\phi}(y_w|x)+\pi^*_{r_\phi}(y_l|x)}$. This motivates the choice model as the following *likelihood function*:

$$\begin{aligned} L(R(\cdot;\phi)) &= \mathbb{E}_{(y_w \succ y_l)}\left[\log \frac{\pi^*_{r_\phi}(y_w|x)}{\pi^*_{r_\phi}(y_w|x) + \pi^*_{r_\phi}(y_l|x)}\right] \\ &= \mathbb{E}_{(y_w \succ y_l)}\left[\log \frac{\pi_{\text{ref}}(y_w|x)\exp(\beta^{-1}r(x,y_w;\phi))}{\pi_{\text{ref}}(y_w|x)\exp(\beta^{-1}r(x,y_w;\phi)) + \pi_{\text{ref}}(y_l|x)\exp(\beta^{-1}r(x,y_l;\phi))}\right]. \end{aligned}$$

When π_{ref} equals to the uniform distribution, this model is equivalent to the BTL model in RLHF:

$$L(R(\cdot;\phi)) = \ell_{\text{RM}}(\phi) := -\mathbb{E}_{(y_w \succ y_l)}\Big[\log\Big(\sigma\big(r(x,y_w;\phi) - r(x,y_l;\phi)\big)\Big)\Big]. \tag{22}$$

Connection to SPIN. There is also an interesting connection between (20) and the SPIN introduced in Sec. IV-A2. Note that in (16), we have shown that the reward difference can be expressed by using the definition of the optimal policy. By plugging (16) into (21a), we obtain the following problem:

$$\begin{aligned} \min_{\phi\in\Phi} \quad & \ell(\phi) := -\mathbb{E}_{(x,y)\sim\mathcal{D}, y'\sim\pi(\cdot|x;\theta_t)}\left[\ell\left(\beta\log\left(\frac{\pi^\star_{r_\phi}(y|x)}{\pi_{\text{ref}}(y|x)}\right) - \beta\log\left(\frac{\pi^\star_{r_\phi}(y'|x)}{\pi_{\text{ref}}(y'|x)}\right)\right)\right] \\ s.t. \quad & \pi^\star_{r_\phi} := \arg\max_{\pi} \quad \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x)}[r(x,y;\phi)] - \beta\mathbb{E}_{x\sim\rho}[D_{\text{KL}}(\pi(\cdot|x)\,\|\pi_{\text{ref}}(\cdot|x))]. \end{aligned}$$

Similarly, as discussed in the DPO approach, one can make two simplifications (1) remove the constraints that the policy has to be optimal w.r.t. some reward; (2) directly parameterize the resulting policy using θ. Then the above problem will be reduced precisely to the SPIN formulation (12).

To further compare the SPIN and IRL, note that at each iteration, once the synthetic preference dataset has been constructed where the preferred data are the demonstration / SFT data and the nonpreferred data are the model-generated data, SPIN directly leverages the synthetic preference dataset and runs DPO to update the policy, while the IRL approach considers two separate steps to update both the reward model and the policy model. Similarly to the comparison of DPO and RLHF, here we expect that IRL enjoys better generalization and can alleviate the distribution shift issue through leveraging online generations in the RL training step, but at the cost of a heavier memory burden and additional online generations.

In summary, the method discussed in this section addressed the question (Q1) we raised at the beginning of this section – instead of directly performing SFT on the demonstration data $\mathcal{D}_{\text{demon}}$, it is also possible to learn a reward function from such a dataset, and we expect that the resulting model will generalize better. Later we will use numerical results to confirm such an advantage.

B. Jointly learning policy and reward from Integrated Human Feedback

Next, let us address question (Q2), by developing an approach that *integrates* different steps of the alignment process. Toward this end, we observe that by leveraging the IRL-based formulation (19), learning from the demonstration data $\mathcal{D}_{\text{demon}}$ becomes very similar to the standard RLHF approach discussed in Section IV-B, which learns preference data $\mathcal{D}_{\text{pref}}$ – both maximize the likelihood of observing the data, while explicitly learning a reward function. Therefore, using reward learning as a common component, it is straightforward to integrate these problems together. In [41] the authors considered the following problem, which extends the IRL problem (19) by adding the BT loss (13):

$$\max_{\phi} \ \ell_{\text{AIHF}}(\phi) := \gamma \mathbb{E}_{x\sim\rho, y\sim\pi^{\text{E}}(\cdot|x)} \left[\log \pi^{\star}_{r_\phi}(y|x)\right] + \alpha \mathbb{E}_{x\sim\rho,(y_l\prec y_w)\sim\pi^{P}(\cdot|x)} \left[\ell\Big(r(x,y_w;\phi) - r(x,y_l;\phi)\Big)\right] \tag{23a}$$

$$\pi^{\star}_{r_\phi} := \arg\max_{\pi} \ \ \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x)}\left[r(x,y;\phi)\right] - \beta \mathbb{E}_{x\sim\rho}[D_{\text{KL}}(\pi(\cdot|x)\,\|\pi_{\text{ref}}(\cdot|x))], \tag{23b}$$

where $\gamma \geq 0$ and $\alpha \geq 0$ are two tuneable parameters. The authors refer to this approach as AIHF (Alignment with Integrated Human Feedback).

To design an algorithm to solve problem (23), one can combine the RLHF Protocol 3 and the IRL algorithm in (21), to arrive at the following iterative algorithm (where t denotes the iteration number):

$$\phi_{t+1} := \arg\min_{\phi} \ -\gamma \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{demo}}, y'\sim\pi(\cdot|x;\theta_t)} \left[\ell\Big(r(x,y;\phi) - r(x,y';\phi)\Big)\right] - \alpha \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{pref}}} \left[\ell\big(r(x,y_w;\phi) - r(x,y_l;\phi)\big)\right], \tag{24a}$$

$$\theta_{t+1} := \arg\max_{\theta} \ \ \mathbb{E}_{x\sim\rho, y\sim\pi(\cdot|x;\theta)}\left[r(x,y;\phi_{t+1})\right] - \beta \mathbb{E}_{x\sim\rho}[D_{\text{KL}}(\pi(\cdot|x;\theta)\,\|\pi_{\text{ref}}(\cdot|x))]. \tag{24b}$$

In fact, compared to the IRL algorithm (21), the only difference is that the reward is learned from *both* preference dataset and the demonstration dataset. In [41], the authors observed that learning reward function from a diverse set of data can be more robust to situations, e.g. when there is an imbalance between the demonstration and preference data, resulting in a more robust policy model; see [41, Sec.3.4] for detailed discussion and a few examples showcasing such an advantage.

Finally, we note that AIHF has several notable special cases. First, if one chooses $\gamma = 0$, then reward learning and policy learning become *completely decoupled*, reducing to the orignal two-step RLHF approach discussed in Section IV-B. Second, similarly to DPO, one can plug in the optimal solution of the lower-level problem (as given in (16)) to the upper-level, remove the lower-level constraint (23b), and directly parameterize the policy. The resulting problem becomes:

$$\max_{\theta} \ \mathbb{E}_{x\sim\rho, y\sim\pi^{\text{E}}(\cdot|x;\theta)}\left[\log \pi(y|x)\right] + \alpha \mathbb{E}_{x\sim\rho,(y_l\prec y_w)\sim\pi^{P}(\cdot|x)} \left[\ell\Big(\beta \log\left(\frac{\pi(y_w|x;\theta)}{\pi_{\text{ref}}(y_w|x)}\right) - \beta \log\left(\frac{\pi(y_l|x;\theta)}{\pi_{\text{ref}}(y_l|x)}\right)\Big)\right].$$

This corresponds to the Regularized Preference Optimization (RPO) approach proposed in [44]. Moreover, if we drop (24b) and apply the DPO trick to (24) to optimize:

$$\max_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{demo}},y'\sim\pi(\cdot|x;\theta_t)}\Big[\ell\Big(\beta\log\left(\frac{\pi(y|x;\theta)}{\pi_{\text{ref}}(y|x)}\right)-\beta\log\left(\frac{\pi(y'|x;\theta)}{\pi_{\text{ref}}(y'|x)}\right)\Big)\Big]$$
$$+\alpha\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\text{pref}}}\Big[\ell\Big(\beta\log\left(\frac{\pi(y_w|x;\theta)}{\pi_{\text{ref}}(y_w|x)}\right)-\beta\log\left(\frac{\pi(y_l|x;\theta)}{\pi_{\text{ref}}(y_l|x)}\right)\Big)\Big].$$

Then it corresponds to the Self-Play with Adversarial Critic (SPAC) approach proposed in [45].

VI. Dataset, Benchmarks, and Evaluation

In this section, we will introduce a few datasets and benchmarks that are often used to evaluate alignment algorithms. Further, we will present some evaluation results comparing a number of algorithms we have discussed so far.

A. Dataset

Typically, datasets used in LLM alignment can be categorized into two types: (1) datasets containing multiple responses for a given prompt, with human (or AI-generated) rankings indicating preference, and (2) datasets containing a single response per prompt, which serve as demonstration datasets.

Preference Dataet. Some of the most well-known preference datasets include: (1) HH-RLHF [1](Anthropic's Helpful-Harmless RLHF), a dataset collected by Anthropic that includes human preference annotations over pairs of responses, focusing on helpfulness (how well the response answers the query) and harmlessness (avoiding toxic, biased, or dangerous content). (2) UltraFeedback [2], which consists of 64K prompts, each accompanied by four model completions from a diverse set of open and proprietary models. Preference labels are provided by GPT-4 based on criteria such as helpfulness and honesty. This dataset has been used to train state-of-the-art chat models such as Zephyr-7B-β.

Demonstration Dataset. Some of the most well-known demonstration datasets include: (1) the TL;DR dataset [3], a dataset collected from Reddit where users provide short summaries (TL;DRs) of their posts. (2) UltraChat [4], an open-source, large-scale, multi-turn dialogue dataset generated using Turbo APIs. It contains 774K training samples and is designed to enhance instruction-following capabilities.

Self-Generated Dataset Besides using public datasets, another line of research employs self-generated datasets for policy or reward optimization. For example, given a prompt, [46] samples one or more responses from a policy π, labels them as preferred or dispreferred, and then updates the policy using

[1]Available at https://huggingface.co/datasets/Anthropic/hh-rlhf.

[2]Available at https://huggingface.co/datasets/openbmb/UltraFeedback.

[3]Available at https://huggingface.co/datasets/CarperAI/openai_summarize_tldr.

[4]Available at https://huggingface.co/datasets/stingning/ultrachat.

the EM algorithm based on these individual preferences. Similarly, [47] also utilizes self-generated data to improve the policy model. This approach can provide more uniform coverage over the entire prompt-response space.

B. Benchmark datasets and benchmark models

AlpacaEval. AlpacaEval [48] is an LLM-based automated evaluation metric – it operates on a fixed set of 805 instructions chosen to be representative of user interactions on the Alpaca web demo. A GPT-4 turbo-based evaluator then compares the responses head-to-head and outputs the probability of preferring the evaluated model. The win rate is then computed as the expected probability that the auto-evaluator prefers the evaluated model's output on the 805 instructions. This win rate serves as a performance measure of the evaluated LM chatbot.

Open LLM Leaderboard. The HuggingFace Open LLM Leaderboard framework [49] is widely adopted for evaluating LLMs. This comprehensive evaluation suite assesses LLM performance across six key tasks: commonsense reasoning, including ARC, HellaSwag, and Winogrande; multi-task language understanding, as measured by MMLU; truthfulness assessment, which evaluates a model's tendency to mimic human falsehoods using TruthfulQA; and mathematical problem-solving, assessed via GSM8K.

MT-Bench. MT-Bench [50] is a benchmark consisting of 80 high-quality multi-turn questions, specifically designed to evaluate multi-turn conversation and instruction-following capabilities in LLMs. It focuses on common real-world use cases while incorporating challenging questions to effectively differentiate model performance. To ensure comprehensive coverage, the benchmark is structured around eight key categories of user prompts: writing, roleplay, extraction, reasoning, mathematics, coding, knowledge I (STEM), and knowledge II (humanities/social sciences).

HELM HELM [51] aims to provide a holistic evaluation of large language models (LLMs). It currently implements a core set of 16 scenarios and 7 metrics. These scenarios—defined as triples of (task, domain, language)—cover six user-facing tasks (e.g., question answering, information retrieval, summarization, toxicity detection), span multiple domains (e.g., news, books), and are currently limited to English. However, they do include a range of English varieties, such as African-American English and regional dialects from different English-speaking countries.

Artificial Analysis[5] Artificial Analysis is an independent AI benchmarking and analysis company. It provides objective evaluations and insights to assist developers, researchers, businesses, and other AI users in selecting the most suitable AI technologies for their specific use cases. The company compares

[5]Available at https://artificialanalysis.ai/leaderboards/models.

and ranks the performance of over 30 large language models (LLMs) across key metrics, including output quality, cost, performance, and speed (measured by tokens per second and time-to-first-token), as well as context window size and other relevant factors.

Reasoning Benchmark. Several challenging benchmarks are also used to evaluate reasoning ability. For example, mathematical benchmarks such as AIME (American Invitational Mathematics Examination), AMC (American Mathematics Competitions), and OlympiadBench assess problem-solving skills in competitive mathematics. Additionally, datasets like the Knights and Knaves (K&K) puzzles and LeetCode are commonly used for evaluating logical reasoning and coding proficiency.

RewardBench. The RewardBench [52] offers a comprehensive set of evaluations for reward models, encompassing key aspects such as chat, instruction-following, coding, safety, and other critical metrics for fine-tuned language models. The RwardBench dataset includes a combination of pre-existing evaluation prompt-completion pairs as well as newly curated examples specifically for this project.

C. Numerical Results

In this subsection, we will present some evaluations results of different alignment approaches. First, we will compare IRL approach (21) with other learning-from-demonstration approaches like SFT (Protocol 1) and SPIN (Protocol 2) to illustrate their differences. Then we will compare a number of algorithms that leverages preference data to further improve the LLM alignment.

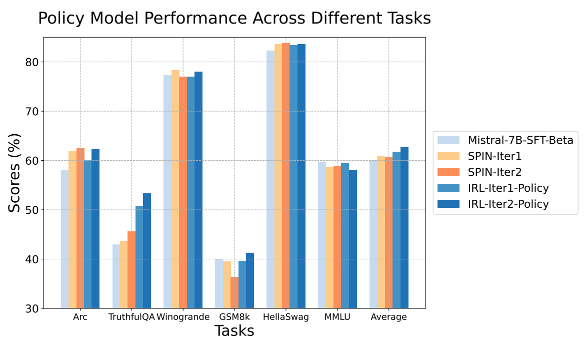
1) Demonstration-only setting: We first present experiments on the UltraChat dataset [53]. We initialize the policy model and reward model from the public checkpoint Mistral-7b-SFT-Beta [6], which is an SFT model fine-tuned from the UltraChat dataset and the base model Mistral-7B-v0.1 [7].

Since UltraChat is a dataset containing only demonstration data, we construct a synthetic preference dataset to train the reward model for the IRL approachs. Specifically, in the reward learning step for each IRL iteration, we treat the demonstration data from UltraChat as the preferred responses and the outputs generated by the IRL policy model as the rejected responses. This approach allows us to create preference pairs without requiring explicit human annotations.

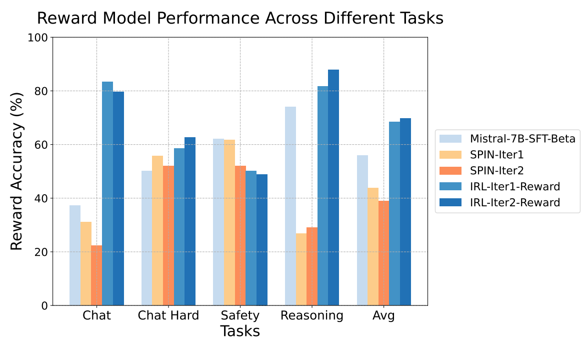
We evaluate reward models obtained by different methods using the RewardBench, assessing performance across various categories relevant to language understanding and generation. For SPIN, since there is no explicit reward model, we evaluate their performance in RewardBench according to the implicit reward expressed as $r(s,a) = \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)}$ where the expression is inspired by (16), and in our experiment π_{ref} is

[6]`HuggingFaceH4/mistral-7b-sft-beta, https://huggingface.co/HuggingFaceH4/mistral-7b-sft-beta`

[7]`mistralai/Mistral-7B-v0.1. https://huggingface.co/mistralai/Mistral-7B-v0.1`



(a) Policy Evaluations on Open LLM Leaderboard



(b) Reward Evaluations on Reward Bench

Fig. 6: Evaluation of policy and reward models.

the SFT model. For SFT model, we evaluate its performance on the RewardBench by using the implicit reward defined as $r(s, a) = \log \pi_{\mathrm{SFT}}(a|s)$. The results, illustrated in Figure 6(b), show that the reward model trained via IRL achieves significant improvements compared to both the base model (initialized from the SFT model) and the implicit reward model extracted from the policy model trained using SPIN [15]. These findings indicate that high-quality demonstration datasets can effectively enhance reward models through leveraging IRL method which can construct synthetic preference pairs through pairing high-quality demonstrations and model generations.

We then evaluate different LLM policy models using the Open LLM Leaderboard [54] and MT-Bench [50]. As shown in Fig 6 and Table III, the IRL-based method outperforms both the mistral-7b-sft-beta checkpoint and SPIN method (running either once or twice of self-generation). These results highlight the potential of leveraging high-quality demonstrations and synthetic preferences to enhance language model performance in dialogue generation tasks.

Tasks	First turn	Second turn	Average
`mistral-7b-sft-beta`	5.66	5.09	5.37
`SPIN-Iter1`	6.75	5.56	6.16
`SPIN-Iter2`	3.18	3.41	3.29
`IRL-Iter1-Policy`	6.71	5.96	6.33
`IRL-Iter2-Policy`	**7.01**	**6.19**	**6.60**

TABLE III: Evaluation of Policy Models in MT-Bench.

2) Aligning with Both Demonstrations and Pairwise Comparisons: In this section, we evaluate a number of alignment algorithm: (1) DPO [55] and IPO [56], (2) SPAC [45], (3) RPO [44], (4) SPIN [15] and (5) AIHF. For this

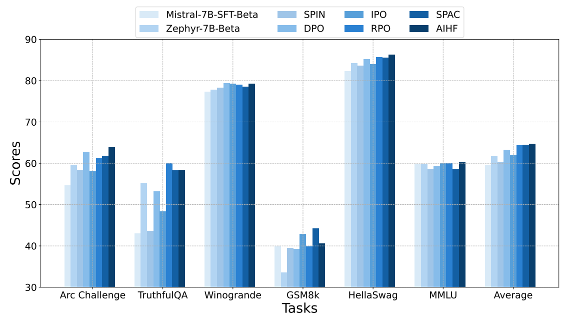


Fig. 7: Evaluations of different alignment approaches across the six benchmark datasets.

Reward Model	Chat	Chat Hard	Safety	Reasoning	Average
DPO Reward Model	37.43%	55.92%	64.14%	47.33%	51.21%
BTL Reward Model	**95.11%**	**56.58%**	63.69%	69.22%	71.15%
AIHF Reward Model	94.41%	55.37%	**63.98%**	**76.75%**	**72.63%**

TABLE IV: Evaluation of Reward Models in Reward-Bench.

experiment, we use UltraFeedback as the preference dataset and UltraChat200k[8] as the demonstration dataset. The base model for our experiments is mistral-7b-sft-beta[9] [57], which is also used to initialize the reward model.

Fig. 7 evaluates the quality of the aligned models trained on different methods. Furthermore, in Table IV, we also evaluate the reward models estimated using different methods (DPO, standard preference learning and AIHF) over the popular benchmark RewardBench [52]. The BTL reward model is trained in the RLHF pipeline using the loss function defined in (13). The DPO reward is obtained by training a DPO loss on preference data (UltraFeedback) and use the following implicit reward $r(s, a) = \log \frac{\pi(a|s)}{\pi_{\text{ref}}(a|s)}$ where π_{ref} is the SFT model. For AIHF, we use the demonstration data UltraChat and generate the non-preferred sample using the base model (mistral-7b-stf-full), then train a reward with UltraFeedback and UltraChat data combined. It can be seen that the reward model estimated by AIHF which incorporates *both* demonstrations and preference achieves better performance, especially on the reasoning tasks.

VII. Concluding Remarks

In this work, we provided an in-depth discussion on aligning LLMs with human feedback, exploring both mathematical foundations and state-of-the-art algorithmic approaches. Despite significant progress in aligning LLMs with human values and preferences, several open challenges remain.

LLM alignment methods continue to face challenges related to scalability and effectiveness. Learning robust and high-quality policies from reward models is challenging and requires extensive human annotation and computation. Existing RL-based fine-tuning methods, such as PPO, demand careful hyperparameter tuning and require a huge amount of compute resources. Moreover, reward hacking remains a significant problem – where models find unintended shortcuts to maximize reward scores without genuinely improving response alignment. Instead of producing responses that align with human intent, models can generate outputs optimized purely for high reward function scores, potentially leading to unreliable behavior. Addressing these concerns requires the development of more efficient RL algorithms, robust and diverse

[8]Available at https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k.

[9]Available at https://huggingface.co/HuggingFaceH4/mistral-7b-sft-beta.

reward functions (e.g., a combination of trained reward functions and the verifiable ones mentioned in Sec. IV-E), and hybrid approaches that integrate supervised learning with RL to improve scalability and alignment fidelity.

Finally, we point out that the SP community has a wealth of expertise that can contribute significantly to alignment research. Given the complexity of aligning LLMs with human preferences, SP methodologies offer valuable tools and frameworks that can improve both the theoretical and practical aspects of alignment. For example, many alignment techniques, including IRL and preference-based optimization discussed in this article, require solving complex inverse problems. SP researchers' expertise in developing effective estimators for these problems, as well as finding effective and performance guaranteed algorithms can help improve reward estimation and enhance alignment robustness. Another potential key contribution is in noise reduction for human feedback data. Human-labeled preference datasets often contain inconsistencies and biases, which can negatively impact alignment performance. Signal processing techniques, such as denoising algorithms, robust statistical modeling, and anomaly detection, can be used to refine and filter noisy preference data. These methods can help improve the quality of preference-based learning by ensuring that reward models are trained on reliable and high-fidelity data. Furthermore, efficient sampling and data compression techniques from SP can enhance the scalability of alignment methods. Many alignment approaches require large-scale datasets and reinforcement learning processes, which are computationally expensive. SP methods for efficient data representation, dimensionality reduction, and signal compression can help reduce computational overhead and improve training efficiency, making alignment methods more feasible at scale. Finally, multi-modal data processing is becoming increasingly relevant as alignment research expands to include models that process text, images, speech, and video. The SP community's experience in multi-modal signal fusion, feature extraction, and cross-modal learning can be instrumental to improve human feedback mechanisms across different modalities.

REFERENCES

[1] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al., "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[2] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al., "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[5] Louis L. Thurstone, “A law of comparative judgment,” *Psychological Review*, vol. 34, no. 4, pp. 273–286, 1927.

[6] Daniel McFadden, “The measurement of urban travel demand,” *Journal of Public Economics*, vol. 3, no. 4, pp. 303–328, 1974.

[7] R. Duncan Luce, *Individual Choice Behavior: A Theoretical analysis*, Wiley, New York, NY, USA, 1959.

[8] Ralph Allan Bradley and Milton E Terry, “Rank analysis of incomplete block designs: I. the method of paired comparisons,” *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[9] Mark J. Machina, “Stochastic choice functions generated from deterministic preferences over lotteries,” *The Economic Journal*, vol. 95, no. 379, pp. 575–594, 1985.

[10] Naftali Tishby and Daniel Polani, “Information theory of decisions and actions,” in *Perception-Action Cycle: Models, Architectures, and Hardware*, Vassilis Cutsuridis, Amir Hussain, and John G. Taylor, Eds., pp. 601–636. Springer New York, New York, NY, 2011.

[11] Fabio Maccheroni, Massimo Marinacci, and Aldo Rustichini, “Ambiguity aversion, robustness, and the variational representation of preferences,” *Econometrica*, vol. 74, no. 6, pp. 1447–1498, 2006.

[12] Siliang Zeng, Mingyi Hong, and Alfredo Garcia, “Structural estimation of markov decision processes in high-dimensional state space with finite-time guarantees,” *arXiv preprint arXiv:2210.01282*, 2022.

[13] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al., “An algorithmic perspective on imitation learning,” *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[14] Dylan J Foster, Adam Block, and Dipendra Misra, “Is behavior cloning all you need? understanding horizon in imitation learning,” *arXiv preprint arXiv:2407.15007*, 2024.

[15] 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.

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

[17] Chris Cundy and Stefano Ermon, “Sequencematch: Imitation learning for autoregressive sequence modelling with backtracking,” *arXiv preprint arXiv:2306.05426*, 2023.

[18] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[20] Ronald J Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, pp. 229–256, 1992.

[21] 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*, vol. 30, 2017.

[22] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Ahmet Üstün, and Sara Hooker, “Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms,” *arXiv preprint arXiv:2402.14740*, 2024.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al., “Training language models to follow instructions with human feedback,” *Advances in neural information processing systems*, vol. 35, pp. 27730–27744, 2022.

[24] 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*, vol. 36, 2024.

[25] Shusheng Xu, Wei Fu, Jiaxuan Gao, Wenjie Ye, Weilin Liu, Zhiyu Mei, Guangju Wang, Chao Yu, and Yi Wu, "Is dpo superior to ppo for llm alignment? a comprehensive study," *arXiv preprint arXiv:2404.10719*, 2024.

[26] 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.

[27] 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," *arXiv preprint arXiv:2402.03300*, 2024.

[28] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al., "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.

[29] 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," *arXiv preprint arXiv:2502.14768*, 2025.

[30] Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong, "Maximum-likelihood inverse reinforcement learning with finite-time guarantees," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10122–10135, 2022.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[32] Luca Viano, Yu-Ting Huang, Parameswaran Kamalaruban, Adrian Weller, and Volkan Cevher, "Robust inverse reinforcement learning under transition dynamics mismatch," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25917–25931, 2021.

[33] Qingyang Wu, Lei Li, and Zhou Yu, "Textgail: Generative adversarial imitation learning for text generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 14067–14075.

[34] Jiaxiang Li, Siliang Zeng, Hoi-To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong, "Getting more juice out of the sft data: Reward learning from human demonstration improves sft for llm alignment," *Advances in Neural Information Processing Systems*, 2024.

[35] Markus Wulfmeier, Michael Bloesch, Nino Vieillard, Arun Ahuja, Jorg Bornschein, Sandy Huang, Artem Sokolov, Matt Barnes, Guillaume Desjardins, Alex Bewley, et al., "Imitating language via scalable inverse reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 37, pp. 90714–90735, 2024.

[36] Siliang Zeng, Yao Liu, Huzefa Rangwala, George Karypis, Mingyi Hong, and Rasool Fakoor, "From demonstrations to rewards: Alignment without explicit human preferences," *arXiv preprint arXiv:2503.13538*, 2025.

[37] Siliang Zeng, Mingyi Hong, and Alfredo Garcia, "Structural estimation of markov decision processes in high-dimensional state space with finite-time guarantees," *Operations Research*, 2024.

[38] Siliang Zeng, Yao Liu, Huzefa Rangwala, George Karypis, Mingyi Hong, and Rasool Fakoor, "From demonstrations to rewards: Alignment without explicit human preferences," 2025.

[39] Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in neural information processing systems*, vol. 31, 2018.

[40] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al., "Zephyr: Direct distillation of lm alignment," *arXiv preprint*

arXiv:2310.16944, 2023.

[41] Chenliang Li, Siliang Zeng, Zeyi Liao, Jiaxiang Li, Dongyeop Kang, Alfredo Garcia, and Mingyi Hong, “Joint demonstration and preference learning improves policy alignment with human feedback,” *arXiv preprint arXiv:2406.06874*, 2024.

[42] Teng Xiao, Yige Yuan, Mingxiao Li, Zhengyu Chen, and Vasant G Honavar, “On a connection between imitation learning and rlhf,” *arXiv preprint arXiv:2503.05079*, 2025.

[43] Drew Fudenberg, Ryota Iijima, and Tomasz Strzalecki, “Stochastic choice and revealed perturbed utility,” *Econometrica*, vol. 83, no. 6, pp. 2371–2409, 2015.

[44] Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang, “Provably mitigating overoptimization in RLHF: Your SFT loss is implicitly an adversarial regularizer,” in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[45] Xiang Ji, Sanjeev Kulkarni, Mengdi Wang, and Tengyang Xie, “Self-play with adversarial critic: Provable and scalable offline alignment for language models,” *arXiv preprint arXiv:2406.04274*, 2024.

[46] Abbas Abdolmaleki, Bilal Piot, Bobak Shahriari, Jost Tobias Springenberg, Tim Hertweck, Rishabh Joshi, Junhyuk Oh, Michael Bloesch, Thomas Lampe, Nicolas Heess, et al., “Preference optimization as probabilistic inference,” *arXiv preprint arXiv:2410.04166*, 2024.

[47] Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang, “Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint,” *arXiv preprint arXiv:2312.11456*, 2023.

[48] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto, “Length-controlled alpacaeval: A simple way to debias automatic evaluators,” *arXiv preprint arXiv:2404.04475*, 2024.

[49] Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf, “Open llm leaderboard,” https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[50] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al., “Judging llm-as-a-judge with mt-bench and chatbot arena,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 46595–46623, 2023.

[51] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Alexander Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew Arad Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda, “Holistic evaluation of language models,” *Transactions on Machine Learning Research*, 2023, Featured Certification, Expert Certification.

[52] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al., “Rewardbench: Evaluating reward models for language modeling,” *arXiv preprint arXiv:2403.13787*, 2024.

[53] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou, “Enhancing chat language models by scaling high-quality instructional conversations,” *arXiv preprint arXiv:2305.14233*, 2023.

[54] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding,

Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou, "A framework for few-shot language model evaluation," 12 2023.

[55] 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.

[56] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello, "A general theoretical paradigm to understand learning from human preferences," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024, pp. 4447–4455.

[57] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al., "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.