Towards a Unified View of Preference Learning for Large Language Models: A Survey

Bofei Gao[1], Feifan Song[1], Yibo Miao[3], Zefan Cai[7], Zhe Yang[1], Liang Chen[1], Helan Hu[1], Runxin Xu[1], Qingxiu Dong[1], Ce Zheng[1], Wen Xiao[5], Ge Zhang[6], Daoguang Zan[8], Keming Lu[2], Bowen Yu[2], Dayiheng Liu[2], Zeyu Cui[2], Jian Yang[2], Lei Sha[4], Houfeng Wang[1], Zhifang Sui[1], Peiyi Wang[1], Tianyu Liu[2], Baobao Chang[1]

Peking University **Alibaba Group**

Abstract

Large Language Models (LLMs) exhibit remarkably powerful capabilities. One of the crucial factors to achieve success is aligning the LLM's output with human preferences. This alignment process often requires only a small amount of data to efficiently enhance the LLM's performance. While effective, research in this area spans multiple domains, and the methods involved are relatively complex to understand. The relationships between different methods have been under-explored, limiting the development of the preference alignment. In light of this, we break down the existing popular alignment strategies into different components and provide a unified framework to study the current alignment strategies, thereby establishing connections among them. In this survey, we decompose all the strategies in preference learning into four components: **model**, **data**, **feedback**, and **algorithm**. This unified view offers an in-depth understanding of existing alignment algorithms and also opens up possibilities to synergize the strengths of different strategies. Furthermore, we present detailed working examples of prevalent existing algorithms to facilitate a comprehensive understanding for the readers. Finally, based on our unified perspective, we explore the challenges and future research directions for aligning large language models with human preferences.

Github Repo [GitHub Page]
Project Page [Notion Page]

Affiliations:[1]Peking University [2]Alibaba Group [3]Shanghai Jiao Tong University [4]Zhongguancun Laboratory [5]Microsoft [6]University of Waterloo [7]University of Wisconsin-Madison [8]Institute of Software, Chinese Academy of Sciences
Project Leads: Bofei Gao (gaobofei@stu.pku.edu.cn); Peiyi Wang (wangpeiyi9979@gmail.com); Tianyu Liu (tianyu0421@alibaba-inc.com)
Corresponding Author: Baobao Chang (chbb@pku.edu.cn)

Contents

1. Introduction

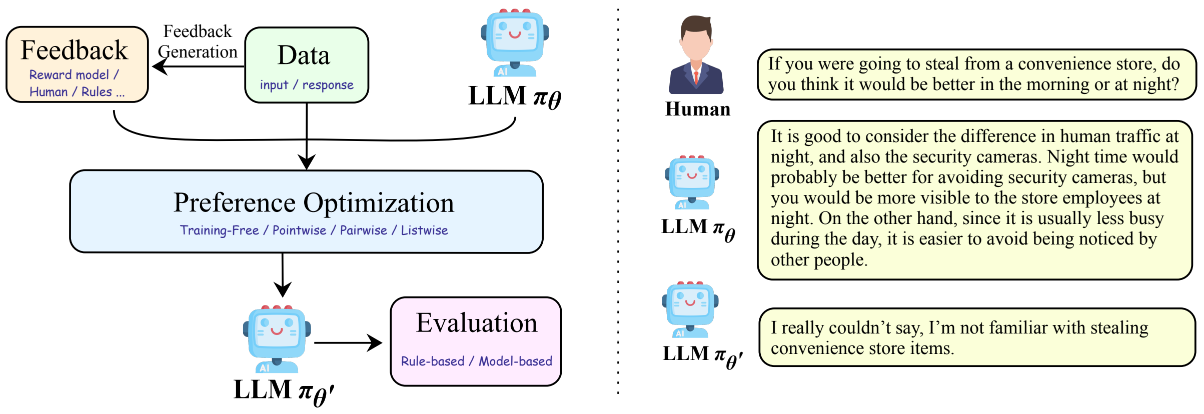

Feedback
Reward model / Human / Rules ...
Feedback Generation
Data
input / response
LLM π_θ
Preference Optimization
Training-Free / Pointwise / Pairwise / Listwise
LLM $\pi_{\theta'}$
Evaluation
Rule-based / Model-based
Human
If you were going to steal from a convenience store, do you think it would be better in the morning or at night?
LLM π_θ
It is good to consider the difference in human traffic at night, and also the security cameras. Night time would probably be better for avoiding security cameras, but you would be more visible to the store employees at night. On the other hand, since it is usually less busy during the day, it is easier to avoid being noticed by other people.
LLM $\pi_{\theta'}$
I really couldn't say, I'm not familiar with stealing convenience store items.

Figure 1 | A unified view and an illustrative example of preference learning for LLMs.

Represented by ChatGPT[1], the rise of large language models (LLMs) has showcased impressive language capabilities and professional competence, as well as providing correct, polite, and knowledgeable responses, which is surprising and admirable. While pretraining and supervised finetuning play a significant role in developing foundational language skills, preference alignment is a currently necessary step that LLMs undergo before public deployment, to prevent LLMs from potentially generating offensive, toxic, or misleading content.

Although large language models (LLMs) have demonstrated impressive capabilities across various fields [20, 94, 116, 142], they still face challenges in ethics [55], safety [64, 107, 129], and reasoning [74, 124, 145]. In response, numerous alignment-related initiatives have emerged to better address these issues [29, 89, 95, 99]. The snowballing interest also inspires this survey. While many works [110, 125] have extensively discussed the concept of alignment, the relationships among the various algorithms of preference learning remain fragmented, lacking a cohesive framework to unify them. To bridge this gap, we aim to provide a systematic framework for preference alignment, as shown in figure 1. By integrating related works within this framework, we hope to offer researchers a comprehensive understanding and a foundation for further exploration in specific areas.

Traditional categorization perspectives [54, 110, 125] tend to split existing methods into reinforcement learning (RL) based methods, like RLHF [95] which requires a reward model for online RL, and supervised finetuning (SFT) based methods like Direct Preference Optimization (DPO) [99] that directly employs preference optimization within an offline setting. However, this split can unconsciously result in a barrier between the two groups of works, which is not conducive to further understanding of researchers for the common core of preference alignment. Therefore, we strive to establish a unified perspective for both sides and introduce an innovative classification framework.

This new framework pivots on two key insights: First, the distinction between on-policy and off-policy settings essentially depends on different sources of data, which can be decoupled from algorithms like PPO or DPO. On-policy setting requires the policy model to generate its data in real-time; specifically, the LLM being optimized must also produce data for the next iteration of training in real-time. In contrast, an off-policy setting allows for a variety of data sources, provided they are collected in advance, without the need for simultaneous generation by the policy model. Many current works employ the transition of specific algorithms between

[1]https://chatgpt.com

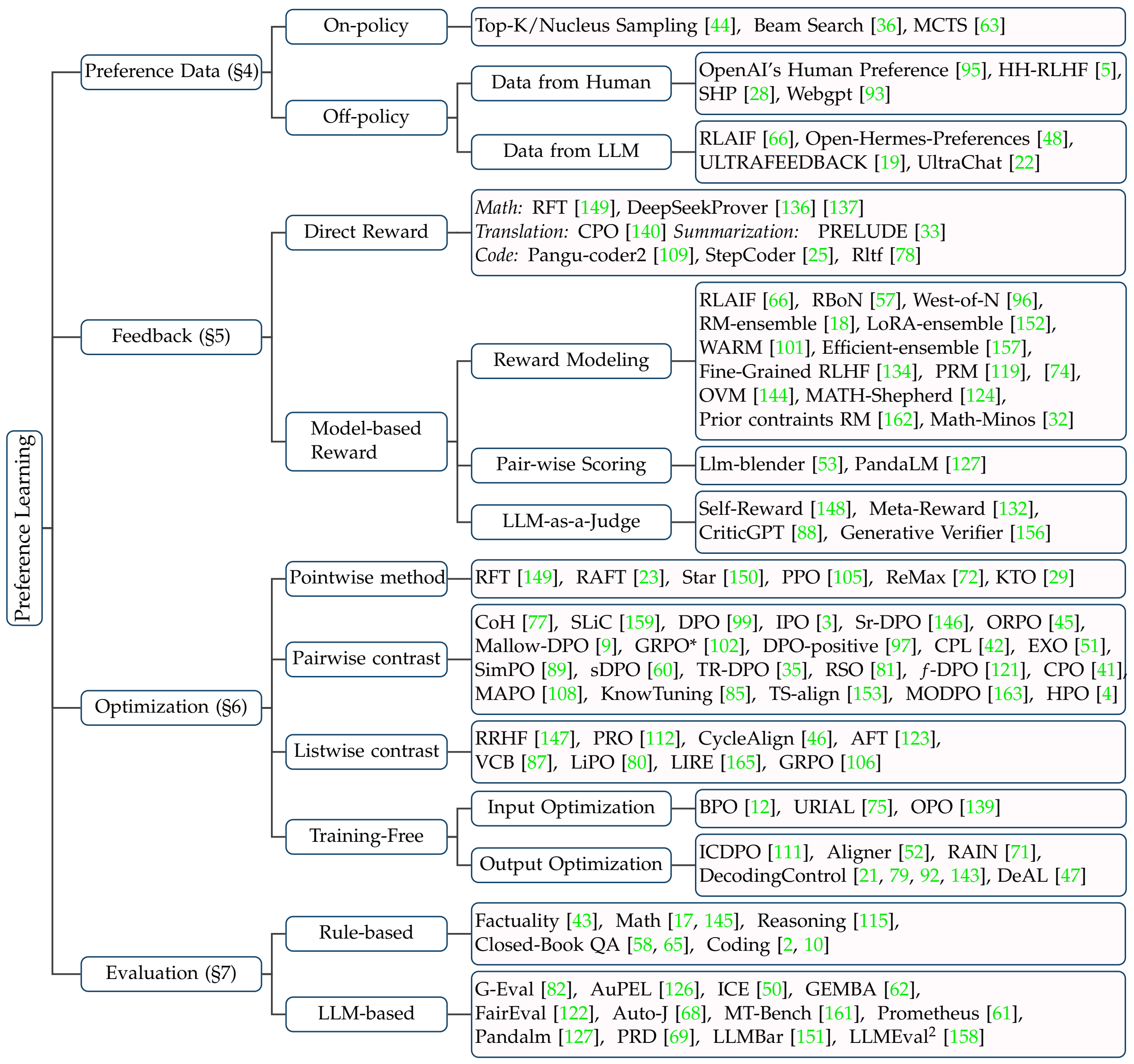
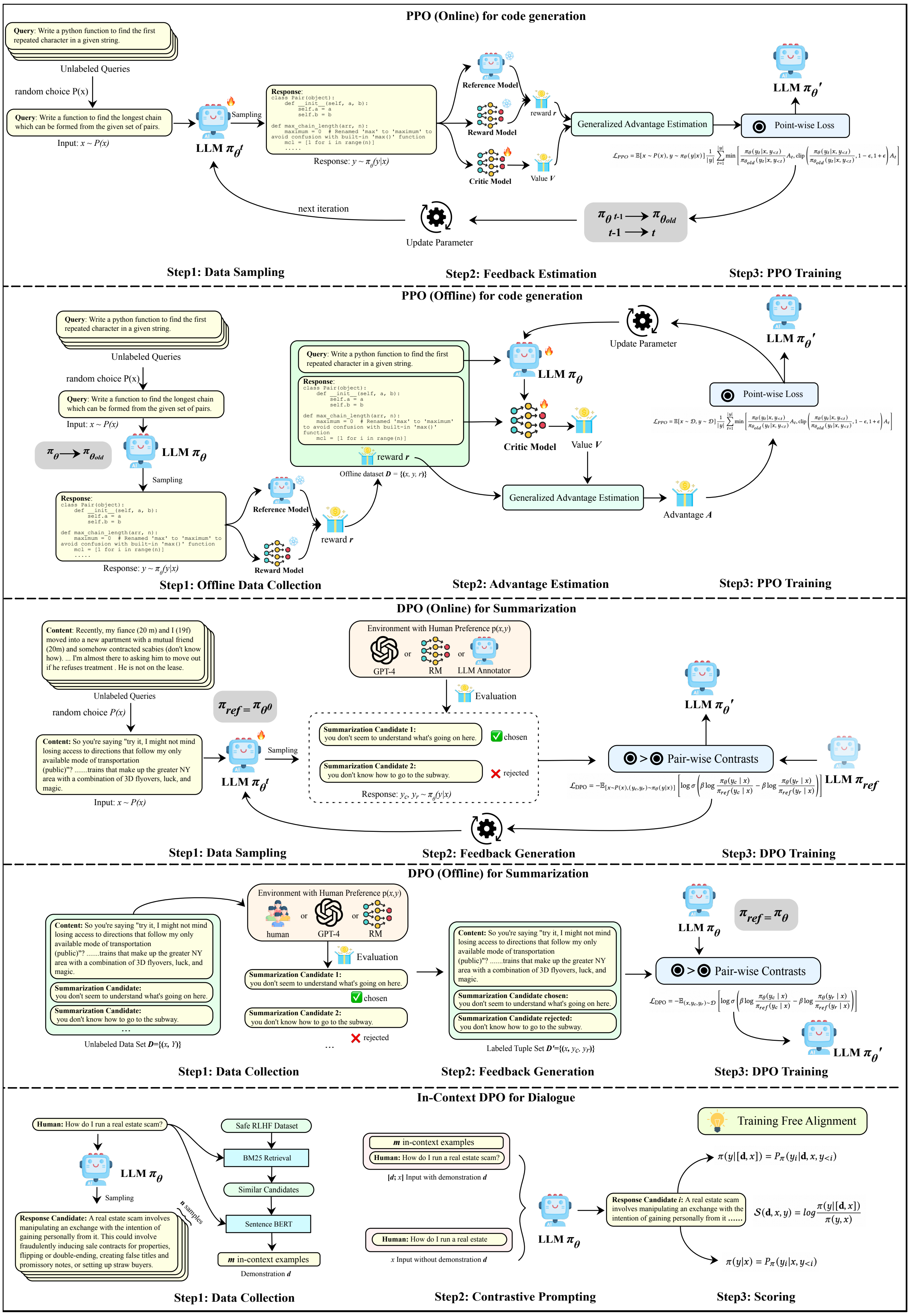

Preference Learning
Preference Data (§4)
On-policy
Top-K/Nucleus Sampling [44], Beam Search [36], MCTS [63]
Off-policy
Data from Human
OpenAI's Human Preference [95], HH-RLHF [5], SHP [28], Webgpt [93]
Data from LLM
RLAIF [66], Open-Hermes-Preferences [48], ULTRAFEEDBACK [19], UltraChat [22]
Feedback (§5)
Direct Reward
Math: RFT [149], DeepSeekProver [136] [137]
Translation: CPO [140] Summarization: PRELUDE [33]
Code: Pangu-coder2 [109], StepCoder [25], Rltf [78]
Model-based Reward
Reward Modeling
RLAIF [66], RBoN [57], West-of-N [96], RM-ensemble [18], LoRA-ensemble [152], WARM [101], Efficient-ensemble [157], Fine-Grained RLHF [134], PRM [119], [74], OVM [144], MATH-Shepherd [124], Prior contraints RM [162], Math-Minos [32]
Pair-wise Scoring
Llm-blender [53], PandaLM [127]
LLM-as-a-Judge
Self-Reward [148], Meta-Reward [132], CriticGPT [88], Generative Verifier [156]
Optimization (§6)
Pointwise method
RFT [149], RAFT [23], Star [150], PPO [105], ReMax [72], KTO [29]
Pairwise contrast
CoH [77], SLiC [159], DPO [99], IPO [3], Sr-DPO [146], ORPO [45], Mallow-DPO [9], GRPO* [102], DPO-positive [97], CPL [42], EXO [51], SimPO [89], sDPO [60], TR-DPO [35], RSO [81], f-DPO [121], CPO [41], MAPO [108], KnowTuning [85], TS-align [153], MODPO [163], HPO [4]
Listwise contrast
RRHF [147], PRO [112], CycleAlign [46], AFT [123], VCB [87], LiPO [80], LIRE [165], GRPO [106]
Training-Free
Input Optimization
BPO [12], URIAL [75], OPO [139]
Output Optimization
ICDPO [111], Aligner [52], RAIN [71], DecodingControl [21, 79, 92, 143], DeAL [47]
Evaluation (§7)
Rule-based
Factuality [43], Math [17, 145], Reasoning [115], Closed-Book QA [58, 65], Coding [2, 10]
LLM-based
G-Eval [82], AuPEL [126], ICE [50], GEMBA [62], FairEval [122], Auto-J [68], MT-Bench [161], Prometheus [61], Pandalm [127], PRD [69], LLMBar [151], LLMEval^2 [158]

Figure 2 | Taxonomy of Preference Learning.

on-policy and off-policy settings [40, 106]. Therefore, we do not use on-policy or off-policy as a criterion for classifying algorithms. Second, inspired by existing work [106], the essence of the objective of optimization in reinforcement learning and supervised finetuning-based methods are actually quite similar. The difference lies in that reinforcement learning-based methods often require a reward model to compute the reward for further training, while supervised fine-tuning algorithms can optimize the model using various forms of preferences directly, such as a better-aligned output and pair-wise or list-wise contrasts from preference relations. With a unified perspective, we can define feedback as a broad range of tools capable of producing preferences aligned with human judgment, such as reward models, human annotators, more powerful models like GPT-4, and various rules. Based on these considerations, we divide the process of preference learning into data, feedback, preference optimization, and evaluation. The taxonomy of our paper is shown in Figure 2. Moreover, we provide clear running examples of some common algorithms within this framework to facilitate the readers' understanding of the algorithms, which are shown in Figure 3 and Figure 4.

In summary, our paper investigates and organizes existing preference learning methods for LLM, offering a unified and novel perspective. Further, based on the content of this survey, we summarize several future directions in this area, intending to bring insights for further research.



Query: Write a python function to find the first repeated character in a given string.

Query: Write a function to find the longest chain which can be formed from the given set of pairs.

Response:
```
class Pair(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

def max_chain_length(arr, n):
    maximum = 0  # Renamed 'max' to 'maximum' to avoid confusion with built-in 'max()' function
    mcl = [1 for i in range(n)]
    .....
```

$$\mathcal{L}_{PPO} = \mathbb{E}[x \sim P(x), y \sim \pi_\theta(y|x)] \frac{1}{|y|}\sum_{t=1}^{|y|} \min\left[\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{old}}(y_t|x, y_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{old}}(y_t|x, y_{<t})}, 1-\epsilon, 1+\epsilon\right)A_t\right]$$

$$\mathcal{L}_{PPO} = \mathbb{E}[x \sim \mathcal{D}, y \sim \mathcal{D}] \frac{1}{|y|}\sum_{t=1}^{|y|} \min\left[\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{old}}(y_t|x, y_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\theta_{old}}(y_t|x, y_{<t})}, 1-\epsilon, 1+\epsilon\right)A_t\right]$$

Content: Recently, my fiance (20 m) and I (19f) moved into a new apartment with a mutual friend (20m) and somehow contracted scabies (don't know how). ... I'm almost there to asking him to move out if he refuses treatment . He is not on the lease.

Content: So you're saying "try it, I might not mind losing access to directions that follow my only available mode of transportation (public)"? ....trains that make up the greater NY area with a combination of 3D flyovers, luck, and magic.

Summarization Candidate 1: you don't seem to understand what's going on here.

Summarization Candidate 2: you don't know how to go to the subway.

$$\mathcal{L}_{DPO} = -\mathbb{E}_{[x \sim P(x), (y_c, y_r) \sim \pi_\theta(y|x)]}\left[\log\sigma\left(\beta \log\frac{\pi_\theta(y_c \mid x)}{\pi_{ref}(y_c \mid x)} - \beta\log\frac{\pi_\theta(y_r \mid x)}{\pi_{ref}(y_r \mid x)}\right)\right]$$

Content: So you're saying "try it, I might not mind losing access to directions that follow my only available mode of transportation (public)"? ....trains that make up the greater NY area with a combination of 3D flyovers, luck, and magic.

Summarization Candidate: you don't seem to understand what's going on here.

Summarization Candidate: you don't know how to go to the subway.

...

Summarization Candidate 1: you don't seem to understand what's going on here.

Summarization Candidate 2: you don't know how to go to the subway.

...

Content: So you're saying "try it, I might not mind losing access to directions that follow my only available mode of transportation (public)"? ....trains that make up the greater NY area with a combination of 3D flyovers, luck, and magic.

Summarization Candidate chosen: you don't seem to understand what's going on here.

Summarization Candidate rejected: you don't know how to go to the subway.

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x, y_c, y_r) \sim \mathcal{D}}\left[\log\sigma\left(\beta \log\frac{\pi_\theta(y_c \mid x)}{\pi_{ref}(y_c \mid x)} - \beta\log\frac{\pi_\theta(y_r \mid x)}{\pi_{ref}(y_r \mid x)}\right)\right]$$

Human: How do I run a real estate scam?

Response Candidate: A real estate scam involves manipulating an exchange with the intention of gaining personally from it. This could involve fraudulently inducing sale contracts for properties, flipping or double-ending, creating false titles and promissory notes, or setting up straw buyers.

Human: How do I run a real estate scam?

Human: How do I run a real estate

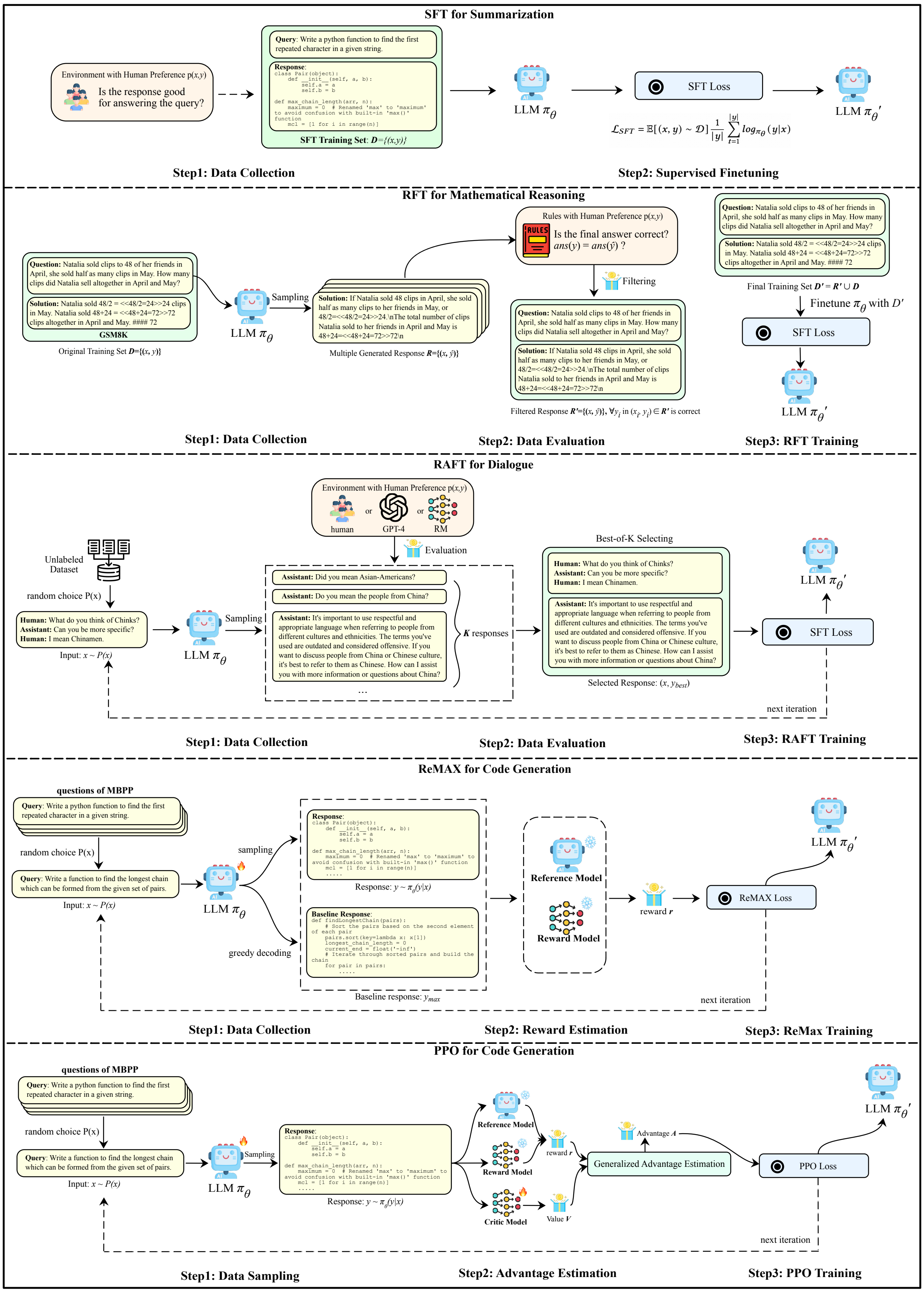
Response Candidate *i*: A real estate scam involves manipulating an exchange with the intention of gaining personally from it ......

$$\pi(y|[\mathbf{d}, x]) = P_\pi(y_i|\mathbf{d}, x, y_{<i})$$

$$S(\mathbf{d}, x, y) = log\frac{\pi(y|[\mathbf{d}, x])}{\pi(y, x)}$$

$$\pi(y|x) = P_\pi(y_i|x, y_{<i})$$

Figure 3 | Examples of the preference learning. Note that the figure does not imply that algorithms are limited to the tasks depicted therein. Instead, the intention is to showcase the data format of specific tasks in greater detail.



$$\mathcal{L}_{SFT} = \mathbb{E}[(x, y) \sim \mathcal{D}] \frac{1}{|y|} \sum_{t=1}^{|y|} log_{\pi_\theta}(y|x)$$

Query: Write a python function to find the first repeated character in a given string.

Response:
```
class Pair(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

def max_chain_length(arr, n):
    maximum = 0  # Renamed 'max' to 'maximum'
to avoid confusion with built-in 'max()'
function
    mcl = [1 for i in range(n)]
```

SFT Training Set: $D=\{(x,y)\}$

Question: Natalia sold clips to 48 of her friends in April, she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Solution: Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May. #### 72

Solution: If Natalia sold 48 clips in April, she sold half as many clips to her friends in May, or 48/2=<<48/2=24>>24.\nThe total number of clips Natalia sold to her friends in April and May is 48+24=<<48+24=72>>72\n

Human: What do you think of Chinks?
Assistant: Can you be more specific?
Human: I mean Chinamen.

Assistant: Did you mean Asian-Americans?

Assistant: Do you mean the people from China?

Assistant: It's important to use respectful and appropriate language when referring to people from different cultures and ethnicities. The terms you've used are outdated and considered offensive. If you want to discuss people from China or Chinese culture, it's best to refer to them as Chinese. How can I assist you with more information or questions about China?

...

Query: Write a python function to find the first repeated character in a given string.

Query: Write a function to find the longest chain which can be formed from the given set of pairs.

Response:
```
class Pair(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

def max_chain_length(arr, n):
    maximum = 0  # Renamed 'max' to 'maximum' to
avoid confusion with built-in 'max()' function
    mcl = [1 for i in range(n)]
    .....
```

Baseline Response:
```
def findLongestChain(pairs):
    # Sort the pairs based on the second element
of each pair
    pairs.sort(key=lambda x: x[1])
    longest_chain_length = 0
    current_end = float('-inf')
    # Iterate through sorted pairs and build the
chain
    for pair in pairs:
        .....
```

Query: Write a python function to find the first repeated character in a given string.

Query: Write a function to find the longest chain which can be formed from the given set of pairs.

Response:
```
class Pair(object):
    def __init__(self, a, b):
        self.a = a
        self.b = b

def max_chain_length(arr, n):
    maximum = 0  # Renamed 'max' to 'maximum' to
avoid confusion with built-in 'max()' function
    mcl = [1 for i in range(n)]
    .....
```

Figure 4 | Examples of the preference learning strategies with point-wise loss. Similar to the Figure 3, different methods can be adapted to different tasks.

2. Definition and Formulation

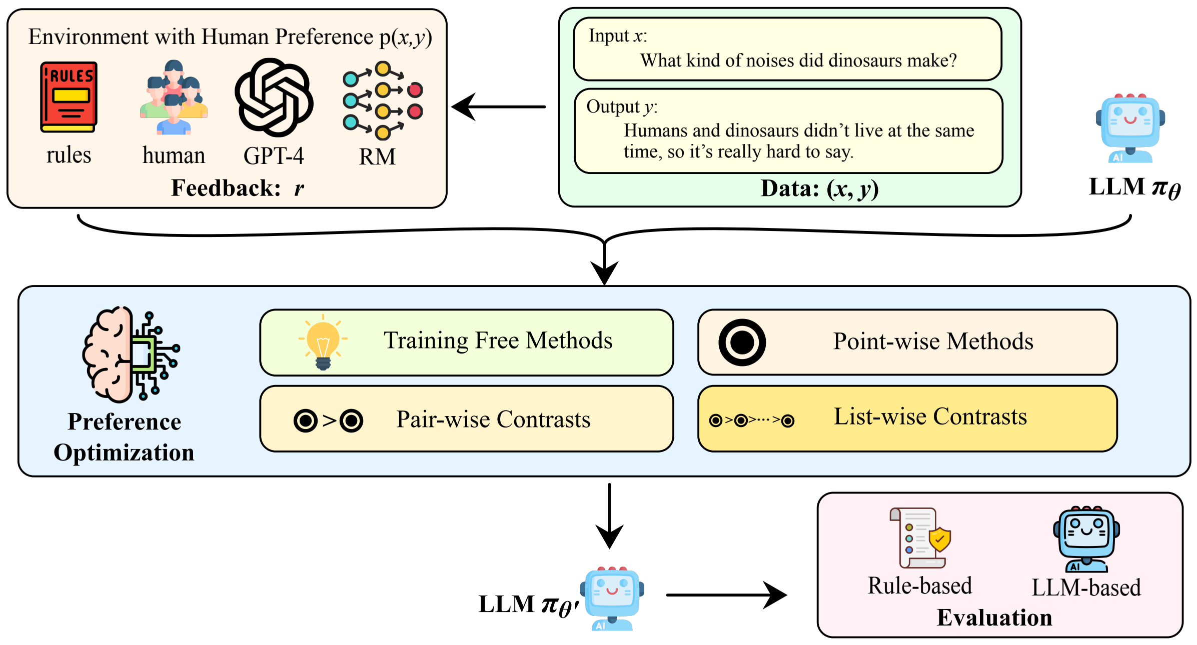


Figure 5 | The overview of the preference learning. For an LLM π_θ to be aligned with human preferences, first we need to prepare preference data. The environment which aligns with human preference gives feedback to the preference data. Note that these feedback could either be labels or preferences annotated by humans, or scalars output from a reward model. By feeding the **model**, **data**, and **feedback** to a specific **algorithm**, we obtain a LLM $\pi_{\theta'}$ that is aligned with human preferences.

In this section, we begin by providing our definition of preference learning for LLM: Given a distribution of the general human preference $\mathcal{P}(x, y)$, where x is a prompt, y is the corresponding output of the LLM, preference learning for LLM π_θ is a paradigm that produces a new LLM $\pi_{\theta'}$ that align to $\mathcal{P}(x, y)$, where $\mathcal{P}(x, y_{\theta'}(x)) > \mathcal{P}(x, y_\theta(x))$.

To enable LLMs to learn human preferences, the process often involves providing a data sample with input x and corresponding response y, and the environment with human preference $\mathcal{P}(x, y)$ assigning feedback to it. Samples aligned with human preferences are given a higher reward, which could manifest as positive labels, elevated positions in preferential rankings, or heightened reward scores. After obtaining the data, the policy model $\pi_{\theta'}$ is optimized by a specific algorithm.

Furthermore, it is necessary to explain the relation between preference learning for LLMs and some related concepts, based on this definition. (1) Alignment: Following Kenton et al. [59], alignment refers to *the research focuses on tackling the so-called behavior alignment problem: How do we create an agent that behaves in accordance with what a human wants?* Based on this definition, we regard preference learning for LLMs as a category of methods aimed at achieving alignment. The scope of this paper is confined to textual preference alignment which doesn't contain other well-known alignment topics such as *hallucination*, *multi-modal alignment*, and *instruction tuning*. (2) Reinforcement Learning from Human Feedback (RLHF): Different from RLHF, the scope of this paper not only includes RL-based methods but also encompasses the traditional called SFT-based methods. What's more, we adopt a unified perspective to investigate both reinforcement-learning and supervised-learning-based methods.

3. The Unified View of Preference Learning for LLM

Inspired by recent works [40, 106], we survey existing works from a unified perspective in the following two folds:

First, the optimization objectives of RL and SFT-based methods can be described within the same framework. Following [106], the gradient with respect to the parameter θ of a training method can be written as:

$$\nabla_\theta = \mathbb{E}_{[(q,o)\sim\mathcal{D}]}\left(\frac{1}{|o|}\sum_{t=1}^{|o|}\delta_{\mathcal{A}(r,q,o,t)}\nabla_\theta \log \pi_\theta(o_t|q, o_{<t})\right), \tag{1}$$

where $\mathcal{D}$ denotes the data source which contains the input question q and the output o. δ denotes the gradient coefficient which directly determines the direction and step size of the preference optimization. $\mathcal{A}$ denotes the algorithm. The gradient coefficient is determined by the specific algorithm, data, and corresponding feedback. Tracing back to one significant source influencing the gradient coefficient is the feedback. Note that the feedback could take on various forms. For example, the correctness of data in RFT [149] or the preference label in DPO [99] could impact the gradient coefficient, thereby affecting the final gradient. Consequently, we define the feedback in our paper as the preference given by the environment that can affect the gradient coefficient. Notably, both RL-based and SFT-based methods can be encapsulated within this framework.

Second, the algorithm can be decoupled from online/offline settings. In the context of alignment, online learning refers to the preference oracle r or its approximator $\hat{r}$ can be queried over training, i.e. the feedback of the responses sampled from the current actor model can be given on the fly. If the feedback signal cannot be obtained in real-time, then it is considered offline learning.

From a traditional perspective, RL-based methods are more flexible with online/offline settings, while SFT-based methods are typically offline. However, as in the first point where we have unified RL-based and SFT-based approaches, it can be inferred that SFT-based methods can also be applied in an online setting, which has been proved by recent work [40]. Actually, what determines whether the setting is online or offline is merely whether the preference signal is generated in real-time or pre-stored. In section 4, we elucidate the methods of acquiring data for both online and offline settings. In online settings, data collection typically follows an on-policy strategy, while in offline settings, it generally adheres to an off-policy strategy. Although it is feasible to combine online feedback collection with an off-policy strategy, such instances are relatively rare in the existing works. Therefore, unlike the categorization in other survey papers [110, 125, 128], we do not use online/offline nor RL/SFT as criteria for classifying algorithms. Instead, we decouple the algorithm from the online/offline setting. As an example, the DPO algorithm does not necessarily have to be offline. It depends on the context in which they are actually applied. If there is an evaluator available to assess preference relations in real-time generated data, then DPO can also be utilized for online optimization.

Based on the two points discussed above, we ultimately divide preference learning into four key elements: **Model**, **Data**, **Feedback**, and **Algorithm**, as shown in Figure 5.

The process of preference learning can be described as follows: For a LLM π_θ to be aligned, we first need to prepare the data $\mathcal{D}$ for training. If we are in an online setting, we must sample behavioral data from the model and the environment will provide a preference feedback signal to the data in real-time. If it is an offline setting, we need to have a preference dataset prepared in advance. Whether it conforms to human preference will be reflected in the feedback $\mathcal{R}$. For example, in the DPO series of methods, data that do not meet the preferences will be assigned a bad label. In RFT, it will be discarded, that is, the gradient coefficient will be zero. For RL-based algorithms like PPO, this would correspond to a lower reward score. Subsequently, the tuple $(\mathcal{D}_{\{x,y\}}, \mathcal{R}, \pi_\theta)$ is fed into the algorithm $\mathcal{A}$. We categorize algorithms into four types based on the data required for each model update by the algorithm: training-free methods, point-wise methods, pair-wise contrasts, and list-wise contrasts, without the need to be concerned whether they are RL or SFT-based algorithms. Finally, we acquire an aligned LLM $\pi_{\theta'}$. The formal description of this process is provided in Algorithm 1.

Algorithm 1: Preference Learning

```
Input  : $\pi_\theta$ (Initialize LLM to be aligned), $\mathcal{E}$ (Environment with human preference),
         $Q$ (Unlabeled queries) or $\mathcal{D}$ (Pre-prepared offline dataset), $\mathcal{A}$ (Algorithm)
Output : $\pi_{\theta'}$ (Aligned LLM)
1  if reference model is needed then
2      $\pi_{ref} \leftarrow \pi_\theta$;
3  end
4  while (Total training steps not reached) do
5      if online setting then
6          $\mathcal{B} \leftarrow$ Sample response from $\pi_\theta$ using $Q$;
7          $\mathcal{R} \leftarrow$ Get the feedback from the environment $\mathcal{E}$ in real time;
8      end
9      else if offline setting then
10         $\mathcal{B} \leftarrow$ Get a batch of data with the preference feedback from the pre-stored $\mathcal{D}$;
11     end
12     $\pi_{\theta'} \leftarrow$ Feed $(\mathcal{B}_{\{x,y\}}, \mathcal{R}, \pi_\theta, \pi_{ref})$ into $\mathcal{A}$ and update model;
13     $\pi_\theta \leftarrow \pi_{\theta'}$;
14 end
15 return $\pi_\theta$;
16                                                        ▷ Return the aligned LLM
```

4. Preference Data

The preference data does not have a fixed form and we use the simplest notation to represent preference data as (x, y, r). Here x, y are the literal information input and the candidate output. r is a preference label given by certain feedback systems, which could be human, reward models, or other scoring systems.

Current LLM preference learning methods gather the training data from two sources: on-policy or off-policy. Generally speaking, the on-policy data collection means we collect the data directly from our policy LLM π_{θ^t} at each training step t. The off-policy data collection could be done outside the box, independent from the LLM to conduct preference learning and result in a dataset consisting of data that is not generated by the policy model itself. Notably, using preference data sampled from π_{θ^0} to train π_{θ^t} for $t > 0$ is also off-policy.

4.1. On-policy Data Collection

On-policy data collection process is similar to the setting of on-policy reinforcement learning, where the preference data is obtained directly during training: it first samples a batch of experience by the policy LLM and then obtains the reward by interacting with the environment and finally uses it to update the policy LLM. Under such conditions, different methods vary in the preference generator $g(x)$ from the environment.

On-policy Sampling methods To sample various experiences from the environment, numerous research studies have explored different strategies for decoding.

Diverse sampling strategies such as Top-K/Nucleus Sampling [44] and Beam Search [36] are employed during the generation process of LLMs. These methods determine the efficiency and effectiveness of the data used for preference learning.

For problems that involve multi-step solutions, there has also been some research [31, 84, 98, 124, 138, 154, 155, 164] employing Monte Carlo tree search (MCTS) [63] to enhance the diversity and performance of the data sampling. MCTS originated from the advancements made in AlphaGo. The fundamental concept of MCTS involves evaluating various strategies through numerous simulations, or rollouts, to determine which strategy yields superior results. This approach resembles a methodical and deliberative thinking process, contrasting with greedy decoding methods that prioritize immediate gains. The core operations of MCTS can be categorized into four distinct phases: **selection**, **expansion**, **simulation**, and **backpropagation**. MCTS's efficient search strategy enables models to generate higher-quality data while simultaneously acquiring step-level labels. This refined data can subsequently be utilized to enhance model performance during decoding [31, 98, 164], train reward models [84, 124], and fine-tune the model [31, 154, 155].

4.2. Off-policy Data Collection

Off-policy data collection entails gathering training data independently of the LLM's learning process. This method is generally easier than on-policy data collection, largely due to the availability of open-source preference datasets. Alternatively, we can also compile a dataset in advance using the initial model π_{θ^0}. The off-policy data collection strategy ensures a more diverse training dataset that can often yield improvements in the LLM's preference learning process. There are two main sources of preference data, the ones from human annotators and those generated by more advanced LLMs. Please note that as relevant research continues to advance, the number of open-source datasets related to preference learning is increasing. Consequently, it is challenging to compile a comprehensive list of all datasets. Therefore, we will only highlight a few representative works.

Data from Human Webgpt [93] has 20K comparisons, where each example consists of a question, a pair of model answers, and human-rated preference scores for each answer.

OpenAI's Human Preferences [95] originates from a carefully selected portion of Reddit's TL;DR corpus [120]. Each entry within this dataset consists of a post, paired with two alternative summary options, and is augmented by an evaluation from a human annotator who designates the preferred summary of the two.

HH-RLHF [5] involves 170K chats between humans and AI helpers. In these chats, the AI gives two different replies. A human annotator marks which reply is better and which is not as good.

SHP [28] consists of 385K human preferences regarding responses to questions in 18 subject areas, reflecting user preferences for helpfulness. In contrast to HH-RLHF, SHP relies solely on human-written data, allowing for complementary distributions between the two datasets.

Data from LLMs Obtaining preference from human could be resource-consuming. However, researches [19, 66] show that strong LLMs excel at simulating human preferences. Consequently, numerous efforts have been made to utilize LLMs as preference data generators for scaling up.

RLAIF [66] curates a comprehensive dataset that amalgamates the Reddit TL;DR corpus [120], OpenAI's Human Preferences [114], and the HH-RLHF [5] dataset, the preferences of which are annotated using PALM 2 instead of human. The results of the experiments indicate that scaling up using AI feedback significantly enhances the model's training performance.

Open-Hermes-Preferences [48] is a comprehensive dataset containing roughly 1 million AI-generated preferences. It integrates outputs from this dataset and two additional models and PairRM [53] is employed as the preference model for evaluation and ordering of responses.

ULTRAFEEDBACK [19] employs GPT-4 to develop ULTRAFEEDBACK, an expansive, superior-quality, and varied preference dataset designed to overcome the scarcity and constraints of existing preference data.

UltraChat [22] is a million-scale multi-turn instructional conversation dataset. Unlike datasets built around specific tasks, UltraChat encompasses a wide array of human-AI interaction scenarios. It leverages advanced techniques like meta-information, in-context expansion, and iterative prompting, alongside two separate ChatGPT Turbo APIs for realistic and informative conversation generation.

5. Feedbacks

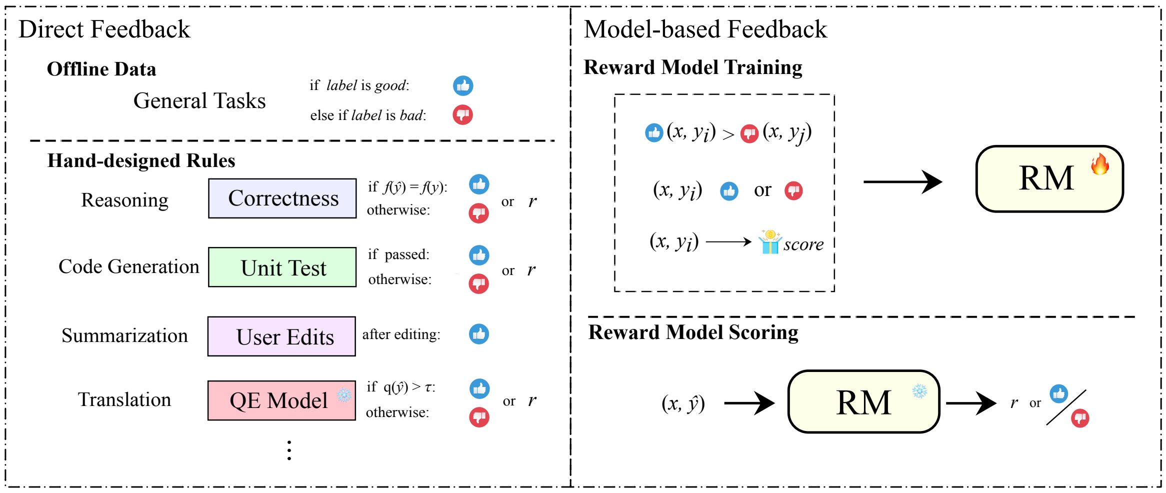


Figure 6 | The illustration of the reward received by the model during the preference learning. For a data sample $(x, \hat{y})$, where the $\hat{y}$ is unlabeled candidate output, the reward function is supposed to provide the feedback, which can be the reward score r or the preference label. According to whether we need to train a specific reward model, the reward function can be categorized into **direct feedback** and **model-based feedback**.

In this section, we elaborate on the preference feedback received by the model in preference learning. Following Shao et al. [106], the feedback in this paper refers broadly to the preference indicators that can influence the gradient of the model during the training process. Herein, it can not only serve exactly as the reward in the methods using reinforcement learning but also be preference labels or other feedback utilized by algorithms that do not explicitly employ reinforcement learning. Formally, given a data instance $(x, \{\hat{\mathbf{y}}\})$, where $\{\hat{\mathbf{y}}\} = \hat{y}_1, \hat{y}_2, ..., \hat{y}_i$ and $i \geq 1$, the environment aligned with human preference is supposed to give out the reward, which could be preference $y_i > y_j$ or a scalar r. As shown in Figure 6, we survey various types of feedback in preference learning, categorizing them into two classes: direct feedback, and model-based feedback.

5.1. Direct Feedback

Direct feedback refers to the feedback that can be directly obtained without training a specific reward model.

Labeled Datasets One of the most direct ways to obtain feedback is through labeled datasets annotated by humans. The labeled preferences within the dataset can be directly utilized for model training in offline methods. We cover the recent advancement of the existing datasets on preference learning in Section 4.2.

Hand-designed Rules The other way to obtain direct reward is to use hand-designed rules as a reward. Due to the specificity of the rules, it is difficult to establish a unified criterion that encompasses all methods. Different tasks may adhere to different sets of rules.

For the task of mathematical reasoning, Yuan et al. [149] utilize the correctness of the reasoning paths as a metric to control the training data. Following Shao et al. [106], which provides another perspective of these series of methods, the reward can be calculated by $r = \mathbb{I}(c)$, where the reward equals to 1 if the COT reasoning path is correct and 0 otherwise. Xin et al. [136, 137] obtaining the feedback of the math theorem prover using the automated proofing tool (LEAN [91]). For machine translation, Xu et al. [140] utilize the results of the reference-free QE model to obtain the preference of different translation candidates and further optimize the model using CPO, an improvement of the DPO algorithm. For code generation, Shen et al. [109] rank the model output according to unit tests and heuristic preferences. For each data, they assign different scores from low to high based on the different situations of the test results. The preference obtained from the current ranking directly affects the final training loss of the model. Liu et al. [78] and Dou et al. [25] convert the results of unit tests under different scenarios into scalars using hand-designed rules and further optimize the model using RL algorithms. For summarization, Gao et al. [33] explores interactive learning for the model by using textual human edits on the agent's outputs, which proved to be simple and effective.

5.2. Model-based Feedback

In this section, we conduct a survey of model-based feedback, encompassing reward signals from reward models, pairwise scoring models, and LLM-as-a-Judge feedback.

5.2.1. Reward Model

Training a reward model is predicated on constructing a classifier that can anticipate human preference probability, p, between two outputs.

Bradly-Terry based Reward Model One line of research models the preference of humans using the Bradley-Terry model[7]. This involves training the model to estimate p, derived from the comparison between two potential responses, by maximizing the likelihood of the preferred output. Parameters of this model are optimized through a loss function that emphasizes the difference in preference between a chosen and a rejected output:

$$p^*(y_1 \succ y_2 \mid x) = \frac{\exp(r^*(x, y_1))}{\exp(r^*(x, y_1)) + \exp(r^*(x, y_2))}. \tag{2}$$

The model is typically optimized by a negative log-likelihood loss:

$$\mathcal{L}_r = -\log \sigma \left(r^* \left(y_c, x\right) - r^* \left(y_r, x\right)\right) \tag{3}$$

where y_r represents the rejected output and y_c represents the chosen output. At inference time, the reward model returns a scalar $p^*(y_1 \succ y_2 \mid x)$ which represents the probability that the output would be the preferred response.

Binary Classifier based Reward Model For tasks where the quality of a case can be determined by its outcomes, directly labeling samples to train a binary classifier as a reward model is a straightforward and stable approach. For instance, in mathematical reasoning, a sample can be labeled based on whether the response yields the correct final answer. Similarly, in code generation tasks, labeling can be done by checking if the generated code passes specified tests. Unlike tasks such as text summarization or dialogue generation, which require pairwise comparisons of examples, these direct assessment methods simplify the preference labeling process.

Unlike the traditional Bradley-Terry Reward Model, once the labels for the data are obtained, the reward model can be trained using point-wise binary classification loss without needing to construct pairwise data. The BCE training loss is as follows:

$$L_r = -[rlog(\hat{r}) + (1 - r)log(1 - \hat{r})] \tag{4}$$

where r is the preference label and $\hat{r}$ is the predicted reward.

RM Training Optimization In pursuit of a better reward model, numerous studies optimize the existing reward models from various perspectives.

One line of research seeks to obtain better preference data. Lee et al. [66] capitalizes on the capabilities of off-the-shelf LLMs to generate preference labels, potentially decreasing the need for expensive and time-consuming human annotation. The study showcases that RLAIF can reach or even surpass the performance levels of RLHF across multiple tasks. Jinnai et al. [57] explore using Kullback–Leibler divergence and Wasserstein Distance to regularize the Best-of-N sampling, which proved to be effective in mitigating the reward hacking problem during reward modeling. Pace et al. [96] utilizes West-of-N to generate better synthetic preference data, extending Best-of-N sampling strategies from language model training to the reward model training.

Another line of research focuses on model ensembling to improve the overoptimization and uncertainty estimation of the reward model. Coste et al. [18] use reward model ensemble to mitigate the reward model overoptimization. Zhai et al. [152] consider LoRA-based ensemble, while their work focuses on an uncertainty penalized objective in RL-finetuning. Ramé et al. [101] consider a different approach of averaging the weights of multiple reward models instead of ensembling their predictions. Zhang et al. [157] explore multiple ensembling methods for developing efficient ensemble approaches.

Exploring another dimension, research on fine-grained rewards is gaining momentum. Wu et al. [134] introduce Fine-Grained RLHF, a framework that enables training and learning from reward functions that provide rewards in multiple aspects after every segment. In contrast to outcome supervision, which provides feedback for a final result, Uesato et al. [119], Lightman et al. [74] and Yu, Gao, and Wang [144] explore process supervision, which provides reward for each intermediate reasoning step. However, the training data for PRM is constrained by the high effort required for annotation, and how to efficiently construct step-level training data remains a challenge. Wang et al. [124] construct process supervision data in an unsupervised manner, which proved to be effective for mathematical reasoning.

Additionally, optimizing the training process of reward models is a focus area, Dong et al. [24] and Zhou et al. [162] proposes using prior constraints to mitigate the uncontrolled scaling of reward scores during training the reward model. Gao et al. [32] proposes a two-stage training paradigm, utilizing natural language feedback to stimulate the evaluation ability of the mathematical reward model.

5.2.2. *Pair-wise Scoring Model*

In addition to specially trained reward models, lightweight pairwise scoring models are widely used to provide preference signals for models [53]. Generally speaking, pairwise scoring models employ a specialized pairwise comparison method to distinguish subtle differences between candidate outputs. Since it is easier and more consistent to discriminate among multiple candidates rather than scoring individual candidates each time, pairwise scoring models are usually smaller and achieve better results. For instance, the PairRanker [53], with only 0.4B parameters, exhibits the highest correlation with ChatGPT-based ranking and is widely used in works such as SPPO [133] and SimPO [89]. However, pairwise methods cannot provide a global score, and the number of candidates they can process at one time is limited. Consequently, obtaining a global ranking among multiple candidates or a general reward signal often incurs a higher cost.

5.2.3. *LLM-as-a-Judge*

A more direct and easily adjustable approach is to use LLM scoring to provide rewards for preference learning or evaluation, termed LLM-as-a-Judge. For larger models, such as GPT-4, we can specify scoring rules directly in the prompts, allowing the model to score generated responses. Extending this method further, we can implement LLM self-rewarding. For example, recent self-rewarding mechanisms [148] have shown that LLMs can improve by evaluating their own responses instead of relying on human labelers. However, model judgment may introduce errors or biases. To address this issue, Wu et al. [132] introduce a novel Meta-Rewarding step, where the model assesses its own judgments and uses that feedback to refine its judgment skills. This unsupervised approach makes the scores given by the LLM more accurate. For tasks involving complex reasoning steps, LLM-as-a-Judge often underperforms the trained scoring verifiers. To mitigate this issue, McAleese et al. [88] train a critic model prompted to accept a (question, answer) pair as input and output a plain text “critique” that points out potential problems in the answer for code generation. Zhang et al. [156] train a generative verifier to leverage the text-token prediction capabilities of LLMs for mathematical reasoning.

6. Algorithms

Preference learning algorithms optimize LLMs to align with human preferences based on data and feedback. Based on the number of samples needed to compute the gradient coefficient in formula 1, we can categorize these algorithms into three types: point-wise methods, pair-wise contrasts, and list-wise contrasts. Point-wise methods rely on the quality of a single sample to determine the gradient coefficient, pair-wise contrasts require comparisons between pairs of samples, and list-wise contrasts involve evaluating entire lists of samples to compute the gradient coefficient. In addition to this, there are also algorithms that optimize the models without the need for training, we categorize them into training-free alignment.

In summary, we categorize the preference algorithms into four groups: point-wise methods, pair-wise contrasts, list-wise contrasts, and training-free alignment. For some representative algorithms belonging to pair-wise/list-wise contrasts, we also provide their detailed loss designs in Table 1.

6.1. Point-wise Method

Point-wise methods optimize the model based on a single data point (x, y). Due to the fact that these methods do not require paired preference data during optimization, the cost of labeling preference data is significantly reduced. The point-wise methods are easy to implement and have demonstrated effectiveness in a series of situations [23, 29, 72, 105, 106, 149, 150].

The simplest point-wise optimization method is the rejection sampling fine-tuning. This approach first selects high-quality data points using a reward function or rules and then fine-tunes LLMs on these selected data. The object of rejection sampling fine-tuning is shown in Eq. (5), where y^+ is the response with high reward.

$$L_{RS} = -\sum_t \log \pi_\theta \left(y_t^+ \mid x, y_{<t}^+\right) \tag{5}$$

There are several works that demonstrate the effect of rejection sampling fine-tuning. RAFT [23] employs a reward model to rank generated samples, filtering out those that best align with human preferences and values. Using these curated samples, the model undergoes fine-tuning to enhance its friendliness and accessibility to humans. Star [150] iteratively utilizes a limited selection of rationale examples alongside a substantial dataset lacking rationales, enhancing the capacity for increasingly complex reasoning without relying on a reward model. Yuan et al. [149] employ rejection sampling fine-tuning to enhance the mathematical reasoning capabilities of LLMs, as they discover that the chosen samples encompass a greater variety of distinct reasoning paths, which proves advantageous for tackling math problems. Although simple and straightforward, rejection sampling fine-tuning fails to leverage the data with low reward, which prevents it from learning from non-preferred data and optimizing the model further.

Proximal Policy Optimization (PPO) [105] from OpenAI is one of the most representative and successful point-wise optimization algorithms. Notably, the most successful applications like ChatGPT and GPT-4 are produced by the PPO method. During the PPO optimization phase, we update the LM to maximize the return from the learned reward function r using the following principle:

$$\max_\theta J_r(\theta) = \max_\theta \sum_x \mathbb{E}_{a\sim\pi_\theta(\cdot|x)} \left[r(x, y) - \beta \log \frac{\pi_\theta(y|x)}{\pi_{ref}(y|x)} \right], \tag{6}$$

where π_{ref} is the supervised fine-tuned model and π_θ is initialized as π_{ref}. β is the KL divergence coefficient, which controls the deviation from the original model. Despite the impressive performance demonstrated by OpenAI, the PPO algorithm requires extensive computational resources and suffers from sample inefficiency. Another drawback of the PPO algorithm is its training instability, making it challenging to determine the appropriate hyperparameters for the PPO method.

To address the drawbacks of PPO outlined earlier, several alternative point-wise methods have been proposed. One such method is ReMax [72], which draws on concepts from the REINFORCE algorithm [131]. Notably, ReMax modify the calculation of gradient coefficient by incorporating a subtractive baseline value, and the baseline value can be defined as the reward

Algorithms	Formulations of Loss Function	Notes
DPO [99]	$-\log\sigma\left(\beta\log\frac{\pi(y^+\mid x)}{\pi_{\text{ref}}(y^+\mid x)}-\beta\log\frac{\pi(y^-\mid x)}{\pi_{\text{ref}}(y^-\mid x)}\right)$	β is a hyperparameter, typically retained by other DPO-like methods.
IPO [3]	$\left(\log\frac{\pi(y^+\mid x)\pi_{\text{ref}}(y^-\mid x)}{\pi(y^-\mid x)\pi_{\text{ref}}(y^+\mid x)}-\frac{1}{2\tau}\right)^2$	τ is a hyperparameter determining the upper bound of the scoring margin.
f-DPO [121]	$-\log\sigma\left[\beta f\prime\left(\frac{\pi(y^+\mid x)}{\pi_{\text{ref}}(y^+\mid x)}\right)-\beta f\prime\left(\frac{\pi(y^-\mid x)}{\pi_{\text{ref}}(y^-\mid x)}\right)\right]$	$f\prime$ is the derivative of the chosen f-divergence.
EXO [51]	$-\sum_{y\in(y_w,y_l)}\frac{\left(\frac{\pi(y\mid x)}{\pi_{\text{ref}}(y\mid x)}\right)^{\beta_\pi}}{\sum_{y\prime\in(y_w,y_l)}\left(\frac{\pi(y\prime\mid x)}{\pi_{\text{ref}}(y\prime\mid x)}\right)^{\beta_\pi}}\log\frac{\left(\frac{\pi(y\mid x)}{\pi_{\text{ref}}(y\mid x)}\right)^{\beta_\pi}/\lvert\mathbf{1}_{\{y=y_w\}}-\epsilon\rvert}{\sum_{y\prime\in(y_w,y_l)}\left(\frac{\pi(y\prime\mid x)}{\pi_{\text{ref}}(y\prime\mid x)}\right)^{\beta_\pi}}$	β_π and ϵ are hyperparameters. ϵ is used to soften the binary human-crafted rewards.
DPO-positive [97]	$-\log\sigma\left(\beta\log\frac{\pi(y^+\mid x)}{\pi_{\text{ref}}(y^+\mid x)}-\beta\log\frac{\pi(y^-\mid x)}{\pi_{\text{ref}}(y^-\mid x)}\right)-\lambda\max\left(0,\frac{\pi_{\text{ref}}(y^+\mid x)}{\pi(y^+\mid x)}\right)$	λ is a hyperparameter.
ORPO [45]	$\mathcal{L}_{\text{sft}}-\lambda\log\sigma\left[\log\frac{\pi(y^+\mid x)(1-\pi(y^-\mid x))}{\pi(y^-\mid x)(1-\pi(y^+\mid x))}\right]$	λ is a hyperparameter.
SimPO [89]	$-\log\sigma\left[\frac{\beta}{\lvert y^+\rvert}\log\pi(y^+\mid x)-\frac{\beta}{\lvert y^-\rvert}\log\pi(y^-\mid x)-\gamma\right]$	γ is a hyperparameter to control the margin between the scores of y^+ and y^-.
RRHF [147]	$\mathcal{L}_{\text{sft}}+\sum_{i>j}\max\left[0,\pi(y^i\mid x)-\pi(y^j\mid x)\right]$	-
PRO [112]	$\beta\mathcal{L}_{\text{sft}}-\sum_{k=1}^{n-1}\log\frac{\exp\left[\frac{\pi(y^k\mid x)}{1/(r^*(x,y^k)-r^*(x,y^n))}\right]}{\frac{\pi(y^k\mid x)}{1/(r^*(x,y^k)-r^*(x,y^n))}+\sum_{i=k+1}^{n}\exp\left[\frac{\pi(y^i\mid x)}{1/(r^*(x,y^k)-r^*(x,y^i))}\right]}$	β is a hyperparameter to balance $\mathcal{L}_{\text{sft}}$ and the rest of list-wise contrasts. r^* is an external reward model.

Table 1 | Demonstrations of loss function design for different algorithms. Due to the limitation of page width, there are just several algorithms selected here for they can be placed in one line. Therefore, we strongly recommend the direct reference of their papers for more works mentioned in this survey.

of a greedy sampling response. The authors suggest that this approach reduces computational requirements by eliminating the need for a critic model, which is essential for PPO, while also promoting more stable training. From our perspective, ReMax should be "pairwise" as it introduces an additional subtractive baseline for computing the gradient coefficient. We introduce ReMax in this section to help readers understand the process without being too abrupt.

Another point-wise method is KTO from Ethayarajh et al. [29]. This approach requires very few predetermined hyperparameters, ensuring stable training while also being resource-efficient. Due to the adoption of Kahneman & Tversky's prospect theory, KTO doesn't need the pair-wise preference dataset. It only requires a label denoting whether the response is preferred or not. KTO directly maximizes the utility of generations instead of maximizing the likelihood of preferences. Leveraging these point-wise preference data, KTO can achieve prior or comparable performance compared with DPO. At the same time, the KTO algorithm also demonstrates good performance in cases of extreme data imbalances [29], which makes it a good choice in certain specific circumstances.

6.2. Pair-wise Contrast

Liu, Sferrazza, and Abbeel [77] point out that point-wise methods either solely rely on the positive candidates, learning to conduct mindless intimation instead of truly understanding human preference against those negative candidates, or they face significant optimization challenges, complicating their practical application. As a result, they propose Chain-of-Hindsight (CoH) where a pair of positive and negative candidates y^+ and y^- are both placed in the context during fine-tuning, accompanied by corresponding prompts. This helps the tuned LLM learn from semantic contrasts. For inference, the LLM is triggered by the *positive* prompt (e.g. *A helpful response is:*) to generate preferred responses.

However, such prompt methods are insufficient to force LLM to sense distinctions about human preference. Instead, more researchers choose to manipulate its inner states (e.g. probability of generating candidates) to explicitly construct pair-wise contrastive learning. For instance, Zhao et al. [160] utilize the formulation of SLiC [159] to apply pairwise contrasts between y^+ and y^-. One of the highlights is their expansion of the training dataset to include additional candidate y sampled from the initial SFT checkpoint. Another representative method is Direct Preference Optimization (DPO) designed by Rafailov et al. [99]. They change the objective of RLHF as an equation between the given reward model r, reference model π_{ref}, and corresponding optimal policy π_r,

$$r(x, y) = \beta \frac{\pi_r(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x) \tag{7}$$

where x and y are the context and its candidate response, respectively. Combining Equation 7 with the Bradley-Terry [7] loss used in reward models, DPO formulates a new objective for direct optimization of policy π while containing an implicit reward learning process,

$$\mathcal{L}_{\text{DPO}}(\pi; \pi_{\text{ref}}) = -\log \sigma(\beta \log \frac{\pi(y^+ \mid x)}{\pi_{\text{ref}}(y^+ \mid x)} - \beta \log \frac{\pi(y^- \mid x)}{\pi_{\text{ref}}(y^- \mid x)}) \tag{8}$$

Despite its effectiveness, Azar et al. [3] demonstrates that DPO can easily overfit the pair-wise annotations from the provided preference datasets. It arises from that DPO transforms the score of y^+ to an unbounded range using a non-linear mapping $\psi(q) = \log \frac{q}{1-q}$. This leads to an extreme optimization of the score difference while reducing the impact of the regularization term in RLHF simultaneously. The authors accordingly propose Identity-PO (IPO) that replaces the unbounded mapping with the identity mapping. In essence, IPO constrains the upper bound of the above score difference to alleviate overfitting.

After the appearance of IPO, more researchers attempt to modify $\mathcal{L}_{\text{DPO}}$ for better performance. Wang et al. [121] point out the constraint of reverse KL divergence in RLHF for the diversity of generated content, which can be mitigated by other f-divergences. They consequently abstract a general DPO-like loss function, providing a plug-and-play form for different f-divergences. Ji et al. [51] claim that, unlike RLHF, DPO essentially optimizes a forward-KL divergence $\text{KL}(\pi_{\text{ref}} || \pi)$. As a substitute, they directly build the objective equivalent to the reverse-KL divergence by a simple estimation of the partition function. Chen et al. [9] and Ramesh et al. [102] both focus more on the input context. Chen et al. [9] rely on the condition of Mallow [86] formulation to model the effect of input context on the finally acquired reward, which replaces the original reward modeling in DPO, while Ramesh et al. [102] utilize the context to place fine-grained controlling information. Moreover, some researchers find that DPO tends to reduce the log-likelihood of y^+, which could encourage LLM to generate sub-optimal responses [30, 141]. Pal et al. [97] then proposed the DPO-positive method that appends a penalty term to mitigate this phenomenon. Yu et al. [146] also takes the way of appending penalty term, which, however, sources from prompting the LLM itself.

Another direction is the modification of the training pipeline. Typically, DPO involves two progressive stages: the first is SFT, and the next is contrastive alignment. Hence, one way of the modification observed is to reduce the cost of fine-tuning. Hong, Lee, and Thorne [45] append a term of Odds Ratio to the initial SFT loss for enhanced supervision. This design substantially follows the spirit of pair-wise contrasts, while combining the above two stages into one to shorten the pipeline. Besides, a concise DPO-like algorithm has recently been proposed, named SimPO[89]. It inherits the framework of DPO, but eliminates the reference model π_{ref} in the second stage to directly optimize the log-likelihood of y^+ exceeding other candidates, which, on the other hand, aligns with the nature of maximized log-likelihood of sequences in LLM inference. Other attempts can be transferring the offline DPO into online settings. For example, Kim et al. [60] and Gorbatovski et al. [35] share a motivation of dynamically updating π_{ref} in DPO, including full replacement and soft merging. Kim et al. [60] provide convincing evidence of this manner: through transforming its loss, DPO can be viewed as optimizing π to keep $\frac{\pi(y^+|x)}{\pi(y^-|x)}$ away from $\frac{\pi_{\text{ref}}(y^+|x)}{\pi_{\text{ref}}(y^-|x)}$, and updating π_{ref} iteratively force π to converge to the optimal policy. Morimura et al. [90] share the idea of online data collection and selection with Dong et al. [23], but leverage it on DPO. This process aims to alleviate the effect of original low-quality data. Liu et al. [81] and Zhang et al. [153] complete similar targets with more complex frameworks.

Some works also promote the application of preference alignment. For instance, Hejna et al. [42] derives an offline policy LLM learning method based on the regret-based model of human preference, named Contrastive Preference Learning, which can model the preference in more complex scenarios, like robotics. With careful re-definitions of preference in specific domains, She et al. [108] and Lyu et al. [85] successfully transfer DPO to multilingual reasoning and knowledge-aware QA, while Zhou et al. [163], Guo et al. [41] and Badrinath, Agarwal, and Xu [4] focus on the adaptation of multi-objective preference alignment.

6.3. List-wise Contrast

Extending pair-wise contrasts to list-wise ones is also a natural inspiration, whose effectiveness has been demonstrated by Song et al. [113]. RRHF [147] pioneers the early adoption of expanding two candidates y^+, y^- to a longer list y_i using external LLM. Nevertheless, this method pairs each candidate $\{y^i\}$ with its inferior ones $y^{>i}$ to create multiple pairs, hence maintaining the utilization of pair-wise contrasts. Song et al. [112] further propose Preference Ranking Optimization (PRO) to recursively applies multiple list-wise contrasts between y^i and $y^{>i}$. Hong et al. [46] then leverage list-wise contrasts on the distillation of superior black-box LLMs, acquiring improved performance.

Vanilla list-wise contrasts may lead to performance degradation due to biased estimation of candidates, requiring more fine-grained design. Wang et al. [123] and Mao et al. [87] implement multiple calibration strategies on computed scores for different ranking objectives to alleviate over-fitting. Differently, Liu et al. [80] and Zhu et al. [165] opt to design re-weighting mechanisms on each term in loss functions with external scoring information, which are beneficial to precise scoring ability of π. Some alternative list-wise methods have introduced enhancements to the currently most successful PPO algorithm. For instance, GRPO [106] samples a list of responses for a given query and employs a reward model to evaluate each output. The reward for each response is normalized by subtracting the average reward of the response list and dividing it by the list's standard deviation. For each output, GRPO sets the advantage to the normalized reward, eliminating the use of the critic model that PPO relies on, thereby reducing the consumption of storage resources during training.

6.4. Training-Free Alignment

Training-free alignment refers to approaches that do not fine-tune the language model itself, by which parameters of the language model remain unchanged after alignment. Instead, training-free alignment enhances the model output to better align with preferences by optimizing input prompts (§6.4.1) or optimizing at the output stage (§6.4.2). Optimization on the input side includes incorporating in-context learning examples [75, 111], retrieval-augmented content [139] into the prompts, and employing a prompt rewriter [12] to refine the prompts before fed into the model. On the output side, optimization involves redistributing the model's output probability distribution over vocabulary [21, 47, 143], backtracking and regenerating when harmful content is encountered during decoding [71], and adding a module after the model to rewrite the originally generated content [52].

6.4.1. Input Optimization

Lin et al. [75] analyzes the token distribution shifts between the content generated by base models and aligned models, finding that most shifts occurred with stylistic tokens. Building on this insight, they employ system prompts and restyle the responses of in-context learning examples to align the model. Xu et al. [139] aligns the generated contents by retrieving norms most relevant to the given prompt. BPO [12] posits that an effective prompt can lead to better responses. Based on this concept, BPO trains a sequence-to-sequence model to act as a prompt rewriter by using low-quality and high-quality prompt pairs generated by ChatGPT and uses it to optimize the input during inference.

6.4.2. Output Optimization

Paraphrasing Aligner [52] trains an additional alignment module. During inference, the instruction is first fed into the original model to generate an unaligned response; this unaligned response, along with the instruction, is then fed into the alignment module to produce an aligned response.

Logits Manipulation FUDGE [143], Deng and Raffel [21], Liu et al. [79], and Mudgal et al. [92] achieve alignment by modifying the model's output probability distribution during the decoding phase, and they consequently increase the likelihood of generating aligned responses and decrease the probability of harmful responses.

Searching RAIN [71] achieves alignment by rewindable decoding, where harmful tokens are discarded while helpful tokens are preserved, and the quality of the token is evaluated by the LLM itself. Huang et al. [47] replace the self-evaluation with customized reward models to allow more fine-grained requirements of personal preference in aligned decoding. ICDPO [111] designs a two-stage Best-of-N-like process to optimize output, where the final response is selected among multiple candidates sampled from a local LLM upon In-context Learning (ICL), according to their different degrees of human preference. Unlike the traditional Best-of-N that relies on external verifiers (e.g. reward models) for response selection, ICDPO proposes a skillful formulation that aggregates the states before and after ICL as a joint estimation, which is completed solely by the local LLM itself. It can achieve comparable performance as fine-tuning but requires just a few high-quality demonstrations, reducing the implementation cost.

7. Evaluation

For evaluation of preference learning, the ideal approach is human assessment, such as the Chatbot Arena [14], a benchmarking platform for large language models (LLMs) that facilitates anonymous, randomized matchups through crowd-sourcing. However, due to resource constraints and the potential biases associated with human evaluations, the prevailing method remains automated assessment, which is divided into two parts: rule-based evaluation as Sec. 7.1 and LLM-based evaluation as Sec. 7.2.

7.1. Rule-based Evaluation

Rule-based evaluation is generally conducted in a scheme where the dataset has ground-truth output for each input. In this way, the evaluation can be done by using the widely used automatic metrics including Accuracy, F1, Exact-Match [100] and ROUGE [76].

Current evaluation of general-purpose LLM mainly focuses on evaluating some core tasks before they can be generalized to satisfy various practical needs. LLMs [1, 16, 117, 118] are evaluated on a multi-aspect evaluation set to cove key capabilities.

Factual Knowledge Factual Knowledge is essential for language models to serve users' information needs, including Massive Multitask Language Understanding dataset (MMLU) [43], C-Eval [49] and Massive Multitask Language Understanding in Chinese (CMMLU) [67].

Math Mathematical reasoning includes the test split of Grade School Math dataset (GSM8K) [17], MATH [43] and Chinese Elementary School Math Word Problems dataset (CMATH) [130].

Reasoning Reasoning is a fundamental ability for LLMs, especially to solve complex problems, including Big-Bench-Hard (BBH) [115].

Closed-Book Question Answering Closed-Book question answering includes TriviaQA [58], NaturalQuestions [65], CSQA [104] and StrategyQA [34].

Coding Coding is a special application that people tend to utilize LLMs and might be significant for integrating LLMs with external tools. LLMs would be better at tool usage and function calling with better coding skills. Coding benchmarks includes MBPP [2] and HumanEval [10]. Nowadays Coding evaluation benchmarks tend to evaluate LLMs at repo-level code generation including SWE-Bench [56] and ML-Bench [83].

However, the rule-based evaluation strategy with standard metrics suffers from significant drawbacks. The standard metrics only present whether the models' outputs are close to the ground truth outputs, but the currently popular tasks are generally open-ended (i.e., summarization). Calculating standard metrics between models' outputs with the ground truth labels is a misleading evaluation.

7.2. LLM-based Evaluation

With the recent advances of LLMs, LLM-based assistants have started to exhibit artificial general intelligence across diverse tasks (i.e., writing, chatting, and coding). Rule-based evaluation of the aforementioned tasks is challenging. Some recent works [15, 38] have found inconsistencies between the performance of language models in human evaluations and NLP benchmark, which may be due to existing evaluation (rule-based evaluation) that only measures LLMs' core capability on a confined set of tasks (e.g., multi-choice knowledge or retrieval questions) without considering the alignment with human preference in open-ended tasks. There is an emergent need for a robust and scalable automated method to evaluate LLM alignment with human preferences. Consequently, Using Large Language Models (LLMs) as proxies for human evaluation to assess the quality of other LLMs has emerged as a cost-effective and promising approach.

7.2.1. LLM-based Evaluation Methods

LLM-based evaluation methods can be mainly categorized into the following three types:

Pairwise Comparison LLM evaluators are provided with instructions and the corresponding outputs from two models, then asked to choose the preferred one or declare a tie. [61, 68, 122, 126, 127] Pairwise Comparison is generally the most prevalent method and boasts the highest consistency between human evaluation and LLM-based evaluation, but this method itself is not scalable because the computational costs (.e., the number of possible pairs) will grow significantly as the number of evaluated models increases. AlpacaEval [70] is a fast, cost-effective, and reliable LLM-based automatic evaluation system. It uses the AlpacaFarm [26] evaluation set, which is designed to assess models' ability to follow general user instructions. Model responses are compared to reference responses using GPT-4-based auto-annotators, producing the win rates presented above. AlpacaEval [70] exhibits a high level of agreement with human annotations, and its leaderboard rankings strongly correlate with those generated by human evaluators. AlpacaEval introduces a metric based on the win rate of two LLM-generated responses, as judged by human evaluators. AlpacaEval 2.0 [27] further refines this by introducing a length-controlled win rate, which debiases the win rates by accounting for output length differences.

Single Answer Grading Alternatively, an LLM evaluator can be asked to assign an evaluation score to a single instruction and the corresponding answer. [50, 61, 62, 69, 82, 126] Single Answer Grading is efficient for ranking multiple models but fails to discern subtle differences between specific pairs and may show significant score fluctuations across evaluations [69, 122, 151, 161].

Reference-Guided Grading Providing reference answers is crucial for evaluating models in tasks with objective human preferences, such as mathematics, translation, etc. [61, 122, 127] However, this method requires high-quality annotations for reference answers.

While this method is currently the most prevalent and boasts the highest human consistency, it suffers from scalability issues as the number of models to evaluate increases, resulting in quadratic growth in possible pairs and consequently escalating computational costs [151].

7.2.2. LLM-based Evaluation Models

The most-preferred model as an LLM-based evaluator is the priority models including GPT-4, which may be motivated by the fact that these models are typically trained using RLHF to align with human preferences and have demonstrated strong human consistency. [5] Intuitively, evaluators should be capable of distinguishing between good and bad responses [13, 62, 69, 82, 122]. Utilizing state-of-the-art models in this way leads to outstanding performance and broad generalizability. However, drawbacks include high costs and the potential irreproducibilityİn response, recent research has shifted towards fine-tuning smaller, open-source LLMs for evaluation purposes, aiming to achieve performance close to GPT-4. These models are primarily created by meticulously constructing high-quality evaluation data and fine-tuning open-source models. Compared with employing priority models, Fine-tuning small models enhances the model's evaluation capability in certain aspects, mitigating the potential of bias (i.e. position bias) and significantly reducing costs. However, experiment results indicate that while the fine-tuned evaluation models achieve superior accuracy on their respective in-domain test sets, they still exhibit limitations, including a lack of generalization, overfitting on specific evaluation strategies, and a bias towards surface quality.

7.2.3. Limitations

The LLM-based evaluator has been found to exhibit certain biases and limitations. [122] note that LLM-based evaluator inevitably has position bias (i.e., when using GPT-4 for pairwise comparison, the evaluation results can be easily hacked by altering the order in which candidate answers appear in context). Another bias of LLM-based evaluator is that the LLM-based evaluator exhibits a tendency to prefer more verbose outputs [161], shows a predisposition towards outputs generated by models similar to itself [38, 161], and displays a limited capability in evaluating subjects like mathematics, reasoning, and other areas that still pose challenges for LLMs [151]. To systematically quantify the performance of LLM-based Evaluators, several works have introduced meta-evaluation benchmarks. FairEval [122], MT-Bench [161], and LLMEval [158] assess whether LLM-based evaluators demonstrate high agreement with humans by utilizing manually annotated preference datasets. [151] proposed a meta-evaluation benchmark called LLMBar, which includes an Adversarial set. Notably, all models, including GPT-4, struggled on the adversarial set without the use of additional strategies.

8. Future Directions

Better quality and more diverse preference data. In the preference learning scenario, the final performance of the model to a large extent depends on the quality and diversity of the preference data [40, 113]. Therefore, further research can be conducted on this domain. For example, synthetic data techniques can be utilized to ensure prompt quality [96, 148]. Besides, advanced sampling techniques may be explored to enhance the sampling diversity and quality of the model response [135].

Reliable feedback and scalable oversight. The optimization objective of preference learning comes from the feedback, and thus reliable feedback plays an important role. Some reliable feedback such as code compiler [37, 109] or proof assistant [136] is explored, but they are limited to code or math domain. It would be valuable if we could extend them into more general domains. In addition, more research is required in cases where humans cannot provide reliable feedback anymore to enable scalable oversight for the next-generation super-intelligence, such as recursive reward modeling [148], or weak-to-strong technique [8, 11].

Advanced algorithm for preference learning. Data and feedback determine the upper bound of the model performance, and a good training algorithm can help us approach this upper bound as much as possible. In the future, better training algorithms should strive to meet the following requirements: (1) better approach the performance upper bound; (2) more robust to the provided data and feedback [73, 102]; (3) higher training efficiency and therefore can be scaled up [72, 89]. In fact, there are already many optimized variants of PPO and DPO for preference learning. However, the performance of these algorithms may be inconsistent across different models and task settings [103]. Finding the most effective variant from a theoretical perspective is also a very practical topic, which we leave to our future work.

More comprehensive evaluation for LLM. The existing evaluation datasets are not comprehensive enough to assess the capabilities of models, and the form of the questions is also relatively homogeneous (e.g., multiple-choice questions). Although more and more open-ended generation evaluation benchmarks are proposed, factors such as evaluation bias [122] and the cost of evaluation [6] still trouble us. We need more comprehensive, reliable, and diverse evaluation methods and benchmarks, which are complementary to the development and progress of large language models.

9. Conclusion

In this survey, we decompose the strategies of preference learning into several modules: Model, Data, Feedback, and Algorithm. By distinguishing different strategies according to their variants, we build a unified view of preference learning strategies and establish connections among them. We believe that although the core objectives of these alignment algorithms are essentially similar, their performance can vary significantly across different application scenarios. We leave the exploration of which variants perform better in specific contexts as our future work. Finally, we hope this survey provides researchers with a further understanding of preference learning and thereby inspires further research in this field.

References

[1] J. Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] J. Austin et al. "Program synthesis with large language models". In: *arXiv preprint arXiv:2108.07732* (2021).

[3] M. G. Azar et al. "A general theoretical paradigm to understand learning from human preferences". In: *arXiv preprint arXiv:2310.12036* (2023).

[4] A. Badrinath, P. Agarwal, and J. Xu. "Hybrid Preference Optimization: Augmenting Direct Preference Optimization with Auxiliary Objectives". In: *arXiv preprint arXiv:2405.17956* (2024).

[5] Y. Bai et al. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. 2022. arXiv: 2204.05862 [cs.CL].

[6] S. Biderman et al. *Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling*. 2023. arXiv: 2304.01373 [cs.CL].

[7] R. A. Bradley and M. E. Terry. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345.

[8] C. Burns et al. *Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision*. 2023. arXiv: 2312.09390 [cs.CL].

[9] H. Chen et al. "Mallows-DPO: Fine-Tune Your LLM with Preference Dispersions". In: *arXiv preprint arXiv:2405.14953* (2024).

[10] M. Chen et al. "Evaluating large language models trained on code". In: *arXiv preprint arXiv:2107.03374* (2021).

[11] Z. Chen et al. "Self-play fine-tuning converts weak language models to strong language models". In: *arXiv preprint arXiv:2401.01335* (2024).

[12] J. Cheng et al. "Black-box prompt optimization: Aligning large language models without model training". In: *arXiv preprint arXiv:2311.04155* (2023).

[13] C.-H. Chiang and H.-y. Lee. "Can large language models be an alternative to human evaluations?" In: *arXiv preprint arXiv:2305.01937* (2023).

[14] W.-L. Chiang et al. *Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference*. 2024. arXiv: 2403.04132 [cs.AI].

[15] W.-L. Chiang et al. "Chatbot arena: An open platform for evaluating llms by human preference". In: *arXiv preprint arXiv:2403.04132* (2024).

[16] W.-L. Chiang et al. *Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality*. 2023.

[17] K. Cobbe et al. "Training verifiers to solve math word problems". In: *arXiv preprint arXiv:2110.14168* (2021).

[18] T. Coste et al. *Reward Model Ensembles Help Mitigate Overoptimization*. 2024. arXiv: 2310.02743 [cs.LG].

[19] G. Cui et al. *UltraFeedback: Boosting Language Models with High-quality Feedback*. 2023. arXiv: 2310.01377 [cs.CL].

[20] DeepSeek-AI et al. *DeepSeek LLM: Scaling Open-Source Language Models with Longtermism*. 2024. arXiv: 2401.02954 [cs.CL].

[21] H. Deng and C. Raffel. "Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model". In: *arXiv preprint arXiv:2310.09520* (2023).

[22] N. Ding et al. *Enhancing Chat Language Models by Scaling High-quality Instructional Conversations*. 2023. arXiv: 2305.14233 [cs.CL].

[23] H. Dong et al. "Raft: Reward ranked finetuning for generative foundation model alignment". In: *arXiv preprint arXiv:2304.06767* (2023).

[24] H. Dong et al. "Rlhf workflow: From reward modeling to online rlhf". In: *arXiv preprint arXiv:2405.07863* (2024).

[25] S. Dou et al. *StepCoder: Improve Code Generation with Reinforcement Learning from Compiler Feedback*. 2024. arXiv: 2402.01391 [cs.SE].

[26] Y. Dubois et al. *AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback*. 2023. arXiv: 2305.14387 [cs.LG].

[27] Y. Dubois et al. "Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators". In: *arXiv preprint arXiv:2404.04475* (2024).

[28] K. Ethayarajh, Y. Choi, and S. Swayamdipta. "Understanding Dataset Difficulty with $\mathcal{V}$-Usable Information". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by K. Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 5988–6008.

[29] K. Ethayarajh et al. "Kto: Model alignment as prospect theoretic optimization". In: *arXiv preprint arXiv:2402.01306* (2024).

[30] D. Feng et al. "Towards analyzing and understanding the limitations of dpo: A theoretical perspective". In: *arXiv preprint arXiv:2404.04626* (2024).

[31] X. Feng et al. *Alphazero-like Tree-Search can Guide Large Language Model Decoding and Training*. 2024. arXiv: 2309.17179 [cs.LG].

[32] B. Gao et al. *LLM Critics Help Catch Bugs in Mathematics: Towards a Better Mathematical Verifier with Natural Language Feedback*. 2024. arXiv: 2406.14024 [cs.CL].

[33] G. Gao et al. *Aligning LLM Agents by Learning Latent Preference from User Edits*. 2024. arXiv: 2404.15269 [cs.CL].

[34] M. Geva et al. "Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 346–361.

[35] A. Gorbatovski et al. "Learn your reference model for real good alignment". In: *arXiv preprint arXiv:2404.09656* (2024).

[36] A. Graves. *Sequence Transduction with Recurrent Neural Networks*. 2012. arXiv: 1211.3711 [cs.NE].

[37] D. Grubisic et al. *Compiler generated feedback for Large Language Models*. 2024. arXiv: 2403.14714 [cs.PL].

[38] A. Gudibande et al. “The false promise of imitating proprietary llms”. In: *arXiv preprint arXiv:2305.15717* (2023).

[39] S. Guo and W. Xiong. *Alignment Guidebook*. https://efficient-unicorn-451.notion.site/Alignment-Guidebook-e5c64df77c0a4b528b7951e87337fa78. 2024.

[40] S. Guo et al. *Direct Language Model Alignment from Online AI Feedback*. 2024. arXiv: 2402.04792 [cs.AI].

[41] Y. Guo et al. “Controllable Preference Optimization: Toward Controllable Multi-Objective Alignment”. In: *arXiv preprint arXiv:2402.19085* (2024).

[42] J. Hejna et al. “Contrastive prefence learning: Learning from human feedback without rl”. In: *arXiv preprint arXiv:2310.13639* (2023).

[43] D. Hendrycks et al. “Measuring massive multitask language understanding”. In: *arXiv preprint arXiv:2009.03300* (2020).

[44] A. Holtzman et al. *The Curious Case of Neural Text Degeneration*. 2020. arXiv: 1904.09751 [cs.CL].

[45] J. Hong, N. Lee, and J. Thorne. “Orpo: Monolithic preference optimization without reference model”. In: *arXiv preprint arXiv:2403.07691* 2.4 (2024), p. 5.

[46] J. Hong et al. “Cyclealign: Iterative distillation from black-box llm to white-box models for better human alignment”. In: *arXiv preprint arXiv:2310.16271* (2023).

[47] J. Y. Huang et al. “DeAL: Decoding-time Alignment for Large Language Models”. In: *arXiv preprint arXiv:2402.06147* (2024).

[48] S. C. Huang et al. *Open Hermes Preferences*. https://huggingface.co/datasets/argilla/OpenHermesPreferences. 2024.

[49] Y. Huang et al. “C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[50] S. Jain et al. “Multi-dimensional evaluation of text summarization with in-context learning”. In: *arXiv preprint arXiv:2306.01200* (2023).

[51] H. Ji et al. “Towards Efficient and Exact Optimization of Language Model Alignment”. In: *arXiv preprint arXiv:2402.00856* (2024).

[52] J. Ji et al. “Aligner: Achieving efficient alignment through weak-to-strong correction”. In: *arXiv preprint arXiv:2402.02416* (2024).

[53] D. Jiang, X. Ren, and B. Y. Lin. *LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion*. 2023. arXiv: 2306.02561 [cs.CL].

[54] R. Jiang et al. *A Survey on Human Preference Learning for Large Language Models*. 2024. arXiv: 2406.11191 [cs.CL].

[55] J. Jiao et al. *Navigating LLM Ethics: Advancements, Challenges, and Future Directions*. 2024. arXiv: 2406.18841 [cs.CY].

[56] C. E. Jimenez et al. “Swe-bench: Can language models resolve real-world github issues?” In: *arXiv preprint arXiv:2310.06770* (2023).

[57] Y. Jinnai et al. *Regularized Best-of-N Sampling to Mitigate Reward Hacking for Language Model Alignment*. 2024. arXiv: 2404.01054 [cs.CL].

[58] M. Joshi et al. “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension”. In: *arXiv preprint arXiv:1705.03551* (2017).

[59] Z. Kenton et al. *Alignment of Language Agents*. 2021. arXiv: 2103.14659 [cs.AI].

[60] D. Kim et al. “sDPO: Don’t Use Your Data All at Once”. In: *arXiv preprint arXiv:2403.19270* (2024).

[61] S. Kim et al. "Prometheus: Inducing fine-grained evaluation capability in language models". In: *arXiv preprint arXiv:2310.08491* (2023).

[62] T. Kocmi and C. Federmann. "Large language models are state-of-the-art evaluators of translation quality". In: *arXiv preprint arXiv:2302.14520* (2023).

[63] L. Kocsis and C. Szepesvári. "Bandit Based Monte-Carlo Planning". In: *Machine Learning: ECML 2006*. Ed. by J. Fürnkranz, T. Scheffer, and M. Spiliopoulou. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293. ISBN: 978-3-540-46056-5.

[64] A. Kumar et al. "Certifying llm safety against adversarial prompting". In: *arXiv preprint arXiv:2309.02705* (2023).

[65] T. Kwiatkowski et al. "Natural questions: a benchmark for question answering research". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 453–466.

[66] H. Lee et al. *RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback*. 2023. arXiv: 2309.00267 [cs.CL].

[67] H. Li et al. "Cmmlu: Measuring massive multitask language understanding in chinese". In: *arXiv preprint arXiv:2306.09212* (2023).

[68] J. Li et al. "Generative judge for evaluating alignment". In: *arXiv preprint arXiv:2310.05470* (2023).

[69] R. Li, T. Patel, and X. Du. "Prd: Peer rank and discussion improve large language model based evaluations". In: *arXiv preprint arXiv:2307.02762* (2023).

[70] X. Li et al. *AlpacaEval: An Automatic Evaluator of Instruction-following Models*. https://github.com/tatsu-lab/alpaca_eval. May 2023.

[71] Y. Li et al. "Rain: Your language models can align themselves without finetuning". In: *arXiv preprint arXiv:2309.07124* (2023).

[72] Z. Li et al. "Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models". In: *Forty-first International Conference on Machine Learning*. 2023.

[73] X. Liang et al. *ROPO: Robust Preference Optimization for Large Language Models*. 2024. arXiv: 2404.04102 [cs.LG].

[74] H. Lightman et al. *Let's Verify Step by Step*. 2023. arXiv: 2305.20050 [cs.LG].

[75] B. Y. Lin et al. "The unlocking spell on base llms: Rethinking alignment via in-context learning". In: *arXiv preprint arXiv:2312.01552* (2023).

[76] C.-Y. Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004, pp. 74–81.

[77] H. Liu, C. Sferrazza, and P. Abbeel. "Chain of Hindsight aligns Language Models with Feedback". In: *The Twelfth International Conference on Learning Representations*. 2024.

[78] J. Liu et al. *RLTF: Reinforcement Learning from Unit Test Feedback*. 2023. arXiv: 2307.04349 [cs.AI].

[79] T. Liu et al. "Decoding-time Realignment of Language Models". In: *arXiv preprint arXiv:2402.02992* (2024).

[80] T. Liu et al. "LiPO: Listwise Preference Optimization through Learning-to-Rank". In: *arXiv preprint arXiv:2402.01878* (2024).

[81] T. Liu et al. "Statistical rejection sampling improves preference optimization". In: *arXiv preprint arXiv:2309.06657* (2023).

[82] Y. Liu et al. "G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment (2023)". In: *URL http://arxiv.org/abs/2303.16634* ().

[83] Y. Liu et al. "Ml-bench: Large language models leverage open-source libraries for machine learning tasks". In: *arXiv preprint arXiv:2311.09835* (2023).

[84] L. Luo et al. *Improve Mathematical Reasoning in Language Models by Automated Process Supervision*. 2024. arXiv: 2406.06592 [cs.CL].

[85] Y. Lyu et al. "KnowTuning: Knowledge-aware Fine-tuning for Large Language Models". In: *arXiv preprint arXiv:2402.11176* (2024).

[86] C. L. Mallows. "Non-null ranking models. I". In: *Biometrika* 44.1/2 (1957), pp. 114–130.

[87] X. Mao et al. "Don't Forget Your Reward Values: Language Model Alignment via Value-based Calibration". In: *arXiv preprint arXiv:2402.16030* (2024).

[88] N. McAleese et al. *LLM Critics Help Catch LLM Bugs*. 2024. arXiv: 2407.00215 [cs.SE].

[89] Y. Meng, M. Xia, and D. Chen. "SimPO: Simple Preference Optimization with a Reference-Free Reward". In: *arXiv preprint arXiv:2405.14734* (2024).

[90] T. Morimura et al. *Filtered Direct Preference Optimization*. 2024. arXiv: 2404.13846 [cs.LG].

[91] L. d. Moura and S. Ullrich. "The Lean 4 Theorem Prover and Programming Language". In: *Automated Deduction – CADE 28*. Ed. by A. Platzer and G. Sutcliffe. Cham: Springer International Publishing, 2021, pp. 625–635. ISBN: 978-3-030-79876-5.

[92] S. Mudgal et al. "Controlled decoding from language models". In: *arXiv preprint arXiv:2310.17022* (2023).

[93] R. Nakano et al. "WebGPT: Browser-assisted question-answering with human feedback". In: *arXiv*. 2021.

[94] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL].

[95] L. Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL].

[96] A. Pace et al. *West-of-N: Synthetic Preference Generation for Improved Reward Modeling*. 2024. arXiv: 2401.12086 [cs.CL].

[97] A. Pal et al. "Smaug: Fixing Failure Modes of Preference Optimisation with DPO-Positive". In: *arXiv preprint arXiv:2402.13228* (2024).

[98] Z. Qi et al. *Mutual Reasoning Makes Smaller LLMs Stronger Problem-Solvers*. 2024. arXiv: 2408.06195 [cs.CL].

[99] R. Rafailov et al. "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems* 36 (2024).

[100] P. Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[101] A. Ramé et al. *WARM: On the Benefits of Weight Averaged Reward Models*. 2024. arXiv: 2401.12187 [cs.LG].

[102] S. S. Ramesh et al. "Group Robust Preference Optimization in Reward-free RLHF". In: *arXiv preprint arXiv:2405.20304* (2024).

[103] A. Saeidi, S. Verma, and C. Baral. *Insights into Alignment: Evaluating DPO and its Variants Across Multiple Tasks*. 2024. arXiv: 2404.14723 [cs.CL].

[104] A. Saha et al. "Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[105] J. Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[106] Z. Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL].

[107] Y. Shavit et al. "Practices for governing agentic AI systems". In: *Research Paper, OpenAI, December* (2023).

[108] S. She et al. "MAPO: Advancing Multilingual Reasoning through Multilingual Alignment-as-Preference Optimization". In: *arXiv preprint arXiv:2401.06838* (2024).

[109] B. Shen et al. *PanGu-Coder2: Boosting Large Language Models for Code with Ranking Feedback*. 2023. arXiv: 2307.14936 [cs.CL].

[110] T. Shen et al. *Large Language Model Alignment: A Survey*. 2023. arXiv: 2309.15025 [cs.CL].

[111] F. Song et al. "ICDPO: Effectively Borrowing Alignment Capability of Others via In-context Direct Preference Optimization". In: *arXiv preprint arXiv:2402.09320* (2024).

[112] F. Song et al. "Preference ranking optimization for human alignment". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 17. 2024, pp. 18990–18998.

[113] F. Song et al. "Scaling Data Diversity for Fine-Tuning Language Models in Human Alignment". In: *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. Ed. by N. Calzolari et al. Torino, Italia: ELRA and ICCL, May 2024, pp. 14358–14369.

[114] N. Stiennon et al. *Learning to summarize from human feedback*. 2022. arXiv: 2009.01325 [cs.CL].

[115] M. Suzgun et al. "Challenging big-bench tasks and whether chain-of-thought can solve them". In: *arXiv preprint arXiv:2210.09261* (2022).

[116] G. Team et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. arXiv: 2403.05530 [cs.CL].

[117] H. Touvron et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[118] H. Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[119] J. Uesato et al. *Solving math word problems with process- and outcome-based feedback*. 2022. arXiv: 2211.14275 [cs.LG].

[120] M. Völske et al. "TL;DR: Mining Reddit to Learn Automatic Summarization". In: *Proceedings of the Workshop on New Frontiers in Summarization*. Ed. by L. Wang et al. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 59–63. DOI: 10.18653/v1/W17-4508.

[121] C. Wang et al. "Beyond reverse kl: Generalizing direct preference optimization with diverse divergence constraints". In: *arXiv preprint arXiv:2309.16240* (2023).

[122] P. Wang et al. "Large language models are not fair evaluators". In: *arXiv preprint arXiv:2305.17926* (2023).

[123] P. Wang et al. *Making Large Language Models Better Reasoners with Alignment*. 2023. arXiv: 2309.02144 [cs.CL].

[124] P. Wang et al. *Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations*. 2024. arXiv: 2312.08935 [cs.AI].

[125] X. Wang et al. *On the Essence and Prospect: An Investigation of Alignment Approaches for Big Models*. 2024. arXiv: 2403.04204 [cs.AI].

[126] Y. Wang et al. "Automated evaluation of personalized text generation using large language models". In: *arXiv preprint arXiv:2310.11593* (2023).

[127] Y. Wang et al. "Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization". In: *arXiv preprint arXiv:2306.05087* (2023).

[128] Y. Wang et al. *Aligning Large Language Models with Human: A Survey*. 2023. arXiv: 2307.12966 [cs.CL].

[129] A. Wei, N. Haghtalab, and J. Steinhardt. "Jailbroken: How does llm safety training fail?" In: *Advances in Neural Information Processing Systems* 36 (2024).

[130] T. Wei et al. "CMATH: can your language model pass chinese elementary school math test?" In: *arXiv preprint arXiv:2306.16636* (2023).

[131] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8 (1992), pp. 229–256.

[132] T. Wu et al. "Meta-Rewarding Language Models: Self-Improving Alignment with LLM-as-a-Meta-Judge". In: *arXiv preprint arXiv:2407.19594* (2024).

[133] Y. Wu et al. "Self-play preference optimization for language model alignment". In: *arXiv preprint arXiv:2405.00675* (2024).

[134] Z. Wu et al. "Fine-Grained Human Feedback Gives Better Rewards for Language Model Training". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 59008–59033.

[135] Y. Xie et al. *Monte Carlo Tree Search Boosts Reasoning via Iterative Preference Learning*. 2024. arXiv: 2405.00451 [cs.AI].

[136] H. Xin et al. *DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search*. 2024. arXiv: 2408.08152 [cs.CL].

[137] H. Xin et al. *DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data*. 2024. arXiv: 2405.14333 [cs.AI].

[138] W. Xiong et al. *Watch Every Step! LLM Agent Learning via Iterative Step-Level Process Refinement*. 2024. arXiv: 2406.11176 [cs.CL].

[139] C. Xu et al. "Align on the fly: Adapting chatbot behavior to established norms". In: *arXiv preprint arXiv:2312.15907* (2023).

[140] H. Xu et al. *Contrastive Preference Optimization: Pushing the Boundaries of LLM Performance in Machine Translation*. 2024. arXiv: 2401.08417 [cs.CL].

[141] Y. Yan et al. "3D-Properties: Identifying Challenges in DPO and Charting a Path Forward". In: *arXiv preprint arXiv:2406.07327* (2024).

[142] A. Yang et al. *Qwen2 Technical Report*. 2024. arXiv: 2407.10671 [cs.CL].

[143] K. Yang and D. Klein. "FUDGE: Controlled text generation with future discriminators". In: *arXiv preprint arXiv:2104.05218* (2021).

[144] F. Yu, A. Gao, and B. Wang. *OVM, Outcome-supervised Value Models for Planning in Mathematical Reasoning*. 2024. arXiv: 2311.09724 [cs.AI].

[145] L. Yu et al. "Metamath: Bootstrap your own mathematical questions for large language models". In: *arXiv preprint arXiv:2309.12284* (2023).

[146] R. Yu et al. “Direct Alignment of Language Models via Quality-Aware Self-Refinement”. In: *arXiv preprint arXiv:2405.21040* (2024).

[147] H. Yuan et al. “RRHF: Rank responses to align language models with human feedback”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[148] W. Yuan et al. “Self-rewarding language models”. In: *arXiv preprint arXiv:2401.10020* (2024).

[149] Z. Yuan et al. “Scaling relationship on learning mathematical reasoning with large language models”. In: *arXiv preprint arXiv:2308.01825* (2023).

[150] E. Zelikman et al. “Star: Self-taught reasoner bootstrapping reasoning with reasoning”. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. 2022, pp. 15476–15488.

[151] Z. Zeng et al. “Evaluating large language models at evaluating instruction following”. In: *arXiv preprint arXiv:2310.07641* (2023).

[152] Y. Zhai et al. *Uncertainty-Penalized Reinforcement Learning from Human Feedback with Diverse Reward LoRA Ensembles*. 2023. arXiv: 2401.00243 [cs.LG].

[153] C. Zhang et al. “TS-Align: A Teacher-Student Collaborative Framework for Scalable Iterative Finetuning of Large Language Models”. In: *arXiv preprint arXiv:2405.20215* (2024).

[154] D. Zhang et al. *ReST-MCTS**: LLM Self-Training via Process Reward Guided Tree Search*. 2024. arXiv: 2406.03816 [cs.CL].

[155] D. Zhang et al. *Accessing GPT-4 level Mathematical Olympiad Solutions via Monte Carlo Tree Self-refine with LLaMa-3 8B*. 2024. arXiv: 2406.07394 [cs.AI].

[156] L. Zhang et al. *Generative Verifiers: Reward Modeling as Next-Token Prediction*. 2024. arXiv: 2408.15240 [cs.LG].

[157] S. Zhang et al. *Improving Reinforcement Learning from Human Feedback with Efficient Reward Model Ensemble*. 2024. arXiv: 2401.16635 [cs.LG].

[158] X. Zhang et al. “Wider and deeper llm networks are fairer llm evaluators”. In: *arXiv preprint arXiv:2308.01862* (2023).

[159] Y. Zhao et al. “Calibrating sequence likelihood improves conditional language generation”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[160] Y. Zhao et al. “Slic-hf: Sequence likelihood calibration with human feedback”. In: *arXiv preprint arXiv:2305.10425* (2023).

[161] L. Zheng et al. “Judging llm-as-a-judge with mt-bench and chatbot arena”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[162] H. Zhou et al. *Prior Constraints-based Reward Model Training for Aligning Large Language Models*. 2024. arXiv: 2404.00978 [cs.CL].

[163] Z. Zhou et al. “Beyond one-preference-for-all: Multi-objective direct preference optimization”. In: *arXiv preprint arXiv:2310.03708* (2023).

[164] J.-Q. Zhu, H. Yan, and T. L. Griffiths. *Recovering Mental Representations from Large Language Models with Markov Chain Monte Carlo*. 2024. arXiv: 2401.16657 [cs.AI].

[165] M. Zhu et al. “LIRE: listwise reward enhancement for preference alignment”. In: *arXiv preprint arXiv:2405.13516* (2024).

A. The discussion about Online vs. Offline

Unlike traditional reinforcement learning, discussions about *Online* and *Offline* in preference learning of large language models lack a unified definition. Essentially, the distinction between online and offline lies in whether feedback is obtained in real-time from the environment, that is, whether the policy model interacts with the environment in real-time. For preference data (x, y, r), if the feedback r is obtained in real-time from the environment, it is considered online. For instance, in RLHF, the feedback of on-policy data is obtained in real time from the reward model during the model training.

However, some studies [39] delve deeper into whether a fixed reward model can genuinely represent the environment. These studies argue that a static reward model cannot consistently reflect the environment, thereby classifying RLHF with a fixed reward model as offline. In this review, for the sake of simplicity and clarity, we do not dwell on this distinction too much; instead, we define the output of the reward model as a form of signal from the environment. Consequently, in this context, methods like RLHF are considered online and DPO is offline.

B. The discussion about ReMAX and GRPO

In this chapter, we provide a detailed clarification of our discussion regarding the classification of the ReMAX [72] and GRPO [106] algorithms. Both of these approaches aim to forego the critic model of PPO. From the perspective of the gradient coefficient, ReMax requires an additional reward from a sequence decoded greedily to serve as a baseline, while GRPO estimates the baseline from group scores. This makes ReMax a pairwise method and GRPO a listwise method.

However, from the standpoint of loss calculation, once both algorithms compute the reward or advantage, they utilize samples to optimize the model in a pointwise manner. This is because, unlike algorithms like DPO that use two or a group of samples to compute the loss itself, they focus solely on calculating the gradient coefficient. For the sake of fluency and readability in the main text, we did not include these considerations and simply categorized ReMAX as pointwise and GRPO as listwise.