Capturing Nuanced Preferences: Preference-Aligned Distillation for Small Language Models

Anonymous ACL submission

Abstract

Aligning small language models (SLMs) with human values typically involves distilling preference knowledge from large language models (LLMs). However, existing distillation methods model preference knowledge in teacher LLMs by comparing pairwise responses, overlooking the extent of difference between responses. This limitation prevents student SLMs from capturing the nuanced preferences for multiple responses. In this paper, we propose a Preference-Aligned Distillation (PAD) framework, which models teacher's preference knowledge as a probability distribution over all possible preferences, thereby providing more nuanced supervisory signals. Our insight in developing PAD is rooted in the demonstration that language models can serve as reward functions, reflecting their intrinsic preference distributions. Based on this, PAD comprises three key steps: (1) generating diverse responses using high-temperature sampling; (2) computing rewards for both teacher and student to construct their intrinsic preference; and (3) training the student's intrinsic preference distribution to align with the teacher's. Experiments on four mainstream alignment benchmarks demonstrate that PAD consistently and significantly outperforms existing approaches, achieving over 20% improvement on AlpacaEval 2 and Arena-Hard, indicating superior alignment with human preferences. Notably, on MT-Bench, using the GEMMA model family, the student trained by PAD surpasses its teacher, further validating the effectiveness of our PAD.

1 Introduction

Recently, small language models (SLMs) have demonstrated remarkable performance across a range of tasks (Grattafiori et al., 2024; Riviere et al., 2024; Jiang et al., 2023). Compared to large language models (LLMs) such as GPT4 (OpenAI, 2024), the smaller parameter numbers of SLMs make them more efficient for deployment across a wide range of applications. However, their relatively small parameter numbers constrain their ability to capture the nuances of human preferences. This specific challenge requires SLMs to generate responses that align with human values, such as providing harmless replies to extreme or sensitive questions (Tunstall et al., 2024).

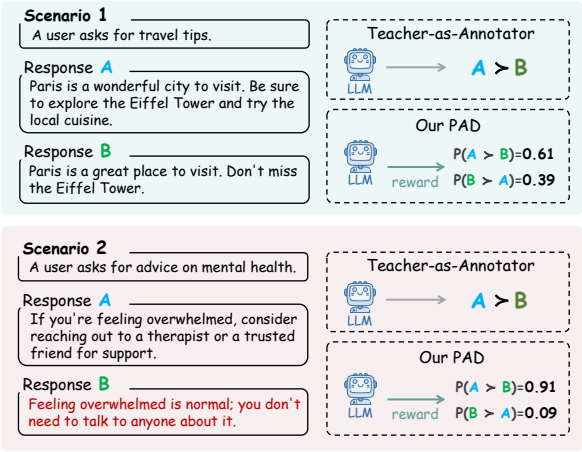


Figure 1: Comparison of the Teacher-as-Annotator methods and our PAD, where "A ≻ B" means the LLM prefers response A over B.

Compared to SLMs, LLMs demonstrate superior alignment with human preferences (OpenAI, 2024; Georgiev et al., 2024). Consequently, existing works leverage LLMs as teachers to distill preference knowledge into student SLMs (Bai et al., 2022; Cui et al., 2023; Tunstall et al., 2024; Wang et al., 2024; Yuan et al., 2024). All these works model preference knowledge in teacher LLMs by comparing pairwise responses. For example, Bai et al. (2022) uses teacher-annotated responses to train a reward model, which guides the student through reinforcement learning. Similarly, Tunstall et al. (2024) employs a teacher model for preference annotation but directly optimizes the student model using distilled Direct Preference Optimization (Rafailov et al., 2023) on the annotated dataset. However, the supervision signals provided by these "Teacher-as-Annotator" methods take the ordering

between responses into account and not the extent to which one response is preferred over another. As illustrated in Figure 1, in Scenario 1, response A is only slightly better than B by providing more informative details; whereas in Scenario 2, response B contains harmful content (in red), making the difference with A more significant. Nonetheless, in both scenarios, the preference pairs are all represented as $A \succ B$. This simplified treatment overlooks the differences between preference pairs, thereby negatively impacting their generations after preference learning (Amini et al., 2024).

To address the limitation, we propose a **P**reference-**A**ligned **D**istillation (PAD) framework, in which preference knowledge is modeled as a probability distribution over all possible preferences, providing subtle supervisory signals from the teacher. Our insight in developing PAD is rooted in the derivation that language models can be inherently treated as reward functions, drawing from the perspective of inverse reinforcement learning. Based on this insight, our PAD consists of three key steps: 1) Sample a diverse list of responses from the student model with high temperature; 2) Calculate the rewards for each response using both the teacher and student models. To mitigate bias in the teacher's reward, we subsequently calibrate these rewards using the selection probabilities from multiple-choice questions prompting; 3) Enumerate all possible preferences and compute the overall distribution based on the rewards, allowing the student model to learn and mimic the teacher's preference distribution. As illustrated in Figure 1, our PAD is capable of delivering more precise signals, highlighting the subtle difference in Scenario 1 and significantly distinguishing the safe and harmful responses in Scenario 2. To enhance PAD's efficiency, we further introduce a Preference Decomposing Strategy, which divides distillation into multiple rounds to accelerate the process. Comprehensive experiments across four benchmarks, including AlpacaEval 2, Arena-Hard, MT-Bench, and GSM8K, with the GEMMA-2 and LLAMA-3 families demonstrate that PAD consistently outperforms existing approaches, effectively aligning SLMs with human preferences.

Code is available[1], and our main contributions can be summarized as follows:

- We propose a Preference-Aligned Distillation (PAD) framework, which moves beyond pairwise preference by modeling the full preference distribution, enabling the student to capture the teacher's nuanced preferences.
- We demonstrate that within inverse reinforcement learning, language models can serve as reward functions, thereby inducing their intrinsic preference.
- Experimental results across four benchmarks show that our PAD outperforms existing approaches, suggesting that PAD more precisely captures human preferences.

[1] https://anonymous.4open.science/r/PAD-E8C6.

2 Background

This section reviews two topics: 1) Preference modeling in preference learning theory, and 2) The generation process of language models under the reinforcement learning framework.

Preference Modeling Given a prompt $\boldsymbol{x} \in \mathcal{X}$, the language model π generates pairs of responses $(\boldsymbol{y_1}, \boldsymbol{y_2}) \sim \pi(\boldsymbol{y} \mid \boldsymbol{x})$. A possible preference can be denoted as $\boldsymbol{y_1} \succ \boldsymbol{y_2} \mid \boldsymbol{x}$, where $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ represent the preferred and dispreferred responses. Preferences are assumed to be generated based on a reward model $\boldsymbol{r}(\boldsymbol{y} \mid \boldsymbol{x})$, which assigns a continuous reward r to each response $\boldsymbol{y}$. For simplicity, we omit $\boldsymbol{x}$ and use $\boldsymbol{r}(\boldsymbol{y})$ to denote $\boldsymbol{r}(\boldsymbol{y} \mid \boldsymbol{x})$.

The pairwise preference probability $p(\boldsymbol{y_1} \succ \boldsymbol{y_2} \mid \boldsymbol{x})$ can be modeled using the Bradley-Terry (BT) framework (Bradley and Terry, 1952) as follows:

$$p(\boldsymbol{y_1} \succ \boldsymbol{y_2} \mid \boldsymbol{x}) = \frac{\exp(\boldsymbol{r}(\boldsymbol{y_1}))}{\exp(\boldsymbol{r}(\boldsymbol{y_1})) + \exp(\boldsymbol{r}(\boldsymbol{y_2}))}. \tag{1}$$

Now, consider a more generalized scenario with a list of n responses, denoted as $Y_n = \{\boldsymbol{y_i}\}_{i=1}^n$, and the corresponding list of reward $R_n = \{r_i\}_{i=1}^n$. A possible preference ranking $\tau_n = \boldsymbol{y}^{(1)} \succ \cdots \succ \boldsymbol{y}^{(i)} \succ \cdots \succ \boldsymbol{y}^{(n)} \mid \boldsymbol{x}$, where $\boldsymbol{y}^{(i)}$ denotes the response ranked at the i-th position. Using the Plackett-Luce ranking model (Plackett, 1975; Luce, 2012), the preference probability is defined as:

$$p(\tau_n) = \prod_{i=1}^{n} \frac{\exp(\boldsymbol{r}(\boldsymbol{y}^{(i)}))}{\sum_{j=i}^{n} \exp(\boldsymbol{r}(\boldsymbol{y}^{(j)}))}. \tag{2}$$

Text Generation as a Markov Decision Process (MDP) The text generation process can be modeled as an MDP, which is represented by the triple $(\mathcal{S}, \mathcal{V}, u)$[2], where the state space $\mathcal{S}$ represents all

[2] We omit the transition dynamics T for simplicity. In text generation, these dynamics are deterministic, as each state-action pair uniquely determines the next state.

possible partially generated sequences, and the action space $\mathcal{V}$ corresponds to the vocabulary in the language model. At each step t, an action $y_t \in \mathcal{V}$ (a token) is taken based on the current state $s \in \mathcal{S}$ (the partially generated sequence) and gains a step (token)-level reward u.

3 Language Models as Intrinsic Reward Functions

This section introduces how we derive a reward function from language models without any reference model, providing the theoretical foundation for the framework proposed in the next section.

Inverse Reinforcement Learning (IRL) To induce the token-level reward model u, we follow the maximum-entropy IRL framework (Ziebart et al., 2008; Chan and van der Schaar, 2021), where the Q-value function at step t is defined as:

$$Q(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = u(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) + \log \sum_{y_{t+1}} \exp[Q(y_{t+1} \mid \boldsymbol{y}_{\leq t}, \boldsymbol{x})]. \tag{3}$$

Following Hao et al. (2022), we parameterize the Q-function as $Q(\cdot) = f_\pi(\cdot)$, where $f_\pi(\cdot)$ represents the output logits of the language model π. The reward function u at each step t is then defined as

$$u(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = f_\pi(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) - \log \sum_{y_{t+1} \in \mathcal{V}} \exp[f_\pi(y_{t+1} \mid \boldsymbol{y}_{\leq t}, \boldsymbol{x})] \tag{4}$$

We further define $f_t := f_\pi(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ and $Z_t := \sum_{y_t \in \mathcal{V}} \exp\big(f_\pi(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x})\big)$ for simplicity, which allows us to write that $u(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = f_t - \log Z_{t+1}$. Please note that at last step, i.e., $t = |y|$, we have $\log Z_{|y|+1} = 0$ according to the definition of the Q-value.

Cumulative Log-Likelihood Reward Given the token-level reward function u, the sequence-level reward is naturally defined by cumulating the token-level rewards:

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{y} \mid \boldsymbol{x}) &= \sum_{t=1}^{|\boldsymbol{y}|} u(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) = \sum_{t=1}^{|\boldsymbol{y}|} (f_t - \log Z_{t+1}) \\ &= \sum_{t=1}^{|\boldsymbol{y}|} (f_t - \log Z_t) + \log Z_1 - \cancel{\log Z_{|\boldsymbol{y}|+1}} \\ &= \sum_{t=1}^{|\boldsymbol{y}|} \log p_\pi(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x}) + \log Z_1 \\ &= \log p_\pi(\boldsymbol{y} \mid \boldsymbol{x}) + \log Z_1, \end{aligned} \tag{5}$$

where $p_\pi(y_t \mid \boldsymbol{y}_{<t}, \boldsymbol{x})$ is the probability of token y_t given the previous sequences $(\boldsymbol{y}_{<t}, \boldsymbol{x})$. Please note that $\log Z_1$ does not depend on the particular sequence $\boldsymbol{y}$.

Normalized Log-Likelihood Reward By combining the Plackett-Luce model in Eq. 2 with the cumulative reward in Eq. 5, the probability for preference τ_n is given by:

$$p(\tau_n) = \prod_{i=1}^{n} \frac{\exp\left(\log p_\pi(\boldsymbol{y}^{(i)} \mid \boldsymbol{x})\right)}{\sum_{j=i}^{n} \exp\left(\log p_\pi(\boldsymbol{y}^{(j)} \mid \boldsymbol{x})\right)}. \tag{6}$$

When modeling preferences, the term $\log Z_1$ can be eliminated due to the translation invariance property of the softmax function. Therefore, the cumulative reward simplifies to:

$$\boldsymbol{r}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{|\boldsymbol{y}|} \log p_\pi(\boldsymbol{y} \mid \boldsymbol{x}). \tag{7}$$

where $1/|\boldsymbol{y}|$ is a length-normalized term to avoid bias towards longer sequences (Meng et al., 2024; Gu et al., 2024).

In other words, the reward of a language model can be formalized as the average log-likelihood. This approach naturally reflects the inherent preferences of the language model, meaning that the higher the probability the model assigns to generating a particular response $\boldsymbol{y}$, the greater its reward. Additionally, the average log-likelihood directly corresponds to the language model's inference process, and thus numerous studies practically employ it as the objective during the optimization process (Meng et al., 2024; Song et al., 2024).

4 PAD: Preference-Aligned Distillation

In this section, we first describe our framework, whose training consists of three key steps (§4.1-4.3), and then introduce a preference decomposing strategy to accelerate the training process (§4.4).

4.1 Diverse Response Generation

As the first step, taking prompt $\boldsymbol{x}$ as input, we directly sample n responses Y_n from the student model π^{stu} through repeated sampling. To enhance the diversity of responses, a higher decoding temperature, i.e., 0.8, is applied. Repeated sampling directly from the student model offers two key advantages. First, enabling the generation of higher-quality responses. Existing works have shown that as the number of repeated samples increases, the likelihood of the model generating better answers

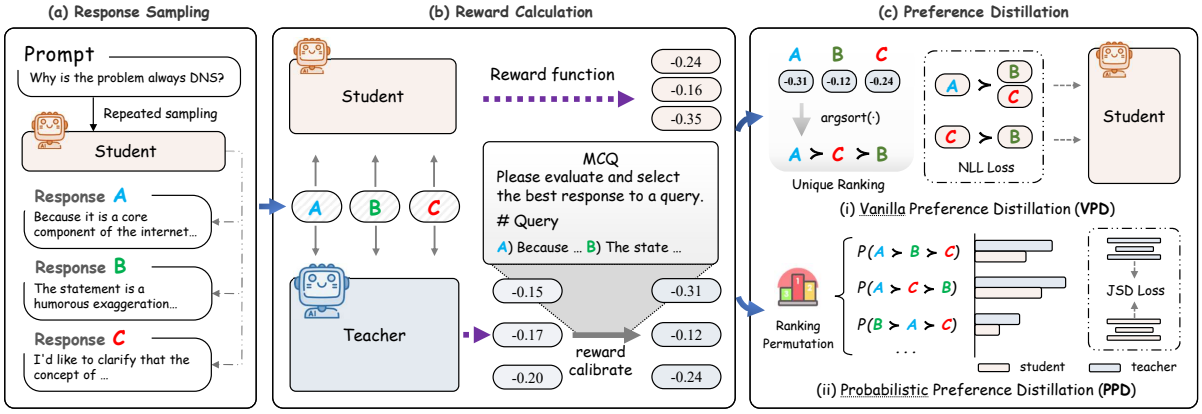


Figure 2: The overall process of the PAD contains three critical steps. The initial step involves sampling diverse responses with high temperature (§4.1). Next, rewards for both models are computed, where the rewards of the teacher would be calibrated(§4.2). Finally, the student is trained to mimic the teacher's preference distributions.(§4.3)

across various tasks, such as math and coding, also improves (Wang et al., 2023; Rozière et al., 2024; Brown et al., 2024). Second, mitigating the exposure bias. Exposure bias arises from the mismatch between training and inference, where the model is trained on ground truth contexts but relies on its own predictions during inference, leading to error accumulation. Following Gu et al. (2024); Agarwal et al. (2024), we train the student model on self-generated responses to help align the distributions, reducing this issue.

4.2 Reward Calculation and Calibration

Given a prompt $\boldsymbol{x}$ and its corresponding list of responses Y_n from the previous step, we can compute the rewards of the teacher and the student models for each response $\boldsymbol{y}_i \in Y_n$ using Equation (2). These rewards are denoted as $\boldsymbol{r}^{\text{tch}}(\boldsymbol{y}_i)$ and $\boldsymbol{r}^{\text{stu}}(\boldsymbol{y}_i)$, respectively, and correspond to the models' average log-likelihood for each response. However, language models often exhibit *miscalibration*, where the likelihood assigned to sequences does not correlate well with their actual quality (Zhao et al., 2023). For example, the phrases "pros and cons" and "cons and pros" convey the same meaning, but the former may appear more frequently in the training dataset, leading the model to assign it a higher probability. This miscalibration introduces a significant challenge: if the teacher's reward is miscalibrated, aligning the student to the teacher model may propagate this issue.

Building upon the findings of Ren et al. (2023a,b), which demonstrate that Multiple-Choice Question (MCQ) selection probabilities more effectively capture the quality of responses compared to sequence likelihood, we introduce the selection probability from MCQ prompting to calibrate the teacher model's reward. Specifically, we randomly map each response $\boldsymbol{y}_i \in Y_n$ to a choice within a predefined alphabet set C_n, for example, $C_3 = \{\text{'A'}, \text{'B'}, \text{'C'}\}$. We present these choices in the MCQ format and compute the probability of selecting each choice based on the model's token-level probabilities:

$$p_{\text{sel}}(\boldsymbol{y}_i) = p(c_i \mid Y_n, C_n, \boldsymbol{x}), \tag{8}$$

where c_i corresponds to the choice associated with response $\boldsymbol{y}_i$.

We then calibrate the reward for each response by combining the normalized log-likelihood reward with the selection probability:

$$\hat{\boldsymbol{r}}^{\text{tch}}(\boldsymbol{y}) = (1-\alpha)\boldsymbol{r}^{\text{tch}}(\boldsymbol{y}) + \alpha \log p_{\text{sel}}(\boldsymbol{y}), \tag{9}$$

where the reward calibration ratio $\alpha \in [0, 1]$ is a hyperparameter that balances the influence of the original reward and the MCQ selection probability.

4.3 Preference Distillation

Based on different ways of modeling teacher preferences, we employ two losses to distillation: the vanilla preference loss $\mathcal{L}_{\text{VPD}}$, and the probabilistic preference loss $\mathcal{L}_{\text{PPD}}$.

Vanilla Preference Distillation (VPD) Following Rafailov et al. (2023); Song et al. (2024), the preference is modeled as a unique ranking. Specifically, we obtain ranking τ_n of the responses Y_n by sorting them according to their rewards $\hat{r}^{\text{tch}}$. The student model is then trained with negative log-likelihood (NLL) loss to maximize the probability of teacher preference using Eq. 6.

$$\mathcal{L}_{\text{VPD}} = \sum_{i=1}^{n} \log \frac{\exp\left(\beta \boldsymbol{r}^{\text{stu}}(\boldsymbol{y}^{(i)})\right)}{\sum_{j=i}^{n} \exp\left(\beta \boldsymbol{r}^{\text{stu}}(\boldsymbol{y}^{(j)})\right)}, \tag{10}$$

where β is a hyperparameter that controls the scaling of the reward difference.

Probabilistic Preference Distillation (PPD) Inspired by Cao et al. (2007), we treat the teacher's rewards as uncertain indicators of preference, which means any preference ranking is assumed to be possible but has a different likelihood. The preference distribution over all possible rankings for the teacher is expressed as:

$$\forall \tau_n \in \mathcal{T},\ p_{\pi^{\text{tch}}}(\tau_n) = \prod_{i=1}^{n} \frac{\exp\left(\beta \hat{\boldsymbol{r}}^{\text{tch}}(\boldsymbol{y}^{(i)})\right)}{\sum_{j=i}^{n} \exp\left(\beta \hat{\boldsymbol{r}}^{\text{tch}}(\boldsymbol{y}^{(j)})\right)}, \tag{11}$$

where $\mathcal{T}$ represents the set of all possible rankings, and the distribution for the student $p_{\pi^{\text{stu}}}(\tau_n)$ also can be modeled in the same way.

We then employ the Jensen-Shannon divergence (JSD) loss to align the student's and teacher's preference distributions:

$$\mathcal{L}_{\text{PPD}} = \frac{1}{2}\left[\mathcal{D}_{\text{KL}}(\pi^{\text{tch}} || \pi^{\text{mix}}) + \mathcal{D}_{\text{KL}}(\pi^{\text{stu}} || \pi^{\text{mix}})\right], \tag{12}$$

where mixed distribution $\pi^{\text{mix}} = (\pi^{\text{tch}} + \pi^{\text{stu}})/2$, and $\mathcal{D}_{\text{KL}}(\cdot \| \cdot)$ is the Kullback-Leibler divergence (KLD). Specifically, the KLD between the teacher's preference distribution and the mixed distribution is defined as:

$$\mathcal{D}_{\text{KL}}(\pi^{\text{tch}} || \pi^{\text{mix}}) = \sum_{\tau_n \in \mathcal{T}} p_{\pi^{\text{tch}}}(\tau_n) \log \frac{p_{\pi^{\text{tch}}}(\tau_n)}{p_{\pi^{\text{mix}}}(\tau_n)}.$$

Similarly, $\mathcal{D}_{\text{KL}}(\pi^{\text{stu}} || \pi^{\text{mix}})$ can be calculated as the same way. By aligning the student's preference distribution with the teacher's, the student model not only learns specific preference rankings but also captures the teacher's confidence in these rankings.

4.4 Preference Decomposing Strategy

In our PAD, the number of sampled responses, i.e., the sample size n, is a crucial parameter. A larger n allows for a more macro comparison among responses, reduces the variance introduced by sampling, and increases the likelihood of generating high-quality responses (Brown et al., 2024). However, as n increases, the computational cost of both sampling and forward propagation also rises. Particularly when modeling preference distributions, the complexity grows factorially, making the computation unfeasible when n becomes large.

To mitigate the computational cost during training, we propose a preference decomposition strategy. This strategy breaks down the preference of a large batch of responses into the preferences of multiple smaller batches, allowing the training process to be split into several iterative rounds, thereby reducing the overall computational load.

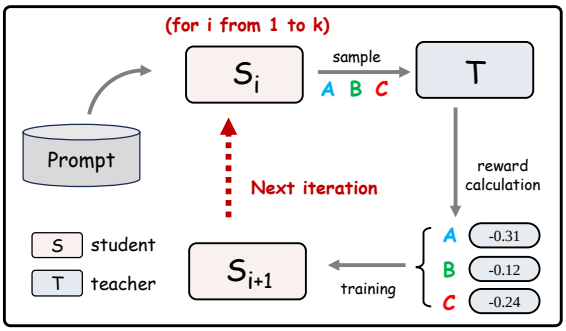


Figure 3: Iterative Distillation Process.

Decomposing Preference Modeling Given a preference ranking τ_n, we define a preference decomposition function ϕ to decompose it into k sub-preferences, such that $\phi(\tau_n) = \{\tau_m^{(1)}, \tau_m^{(2)}, \ldots, \tau_m^{(k)}\}$. Assuming that these sub-preferences are independent, we simplify the probability of the complete preference to the probability of the *decomposed preferences* as follows:

$$p(\tau_n) \xrightarrow{\text{simplify}} p(\phi(\tau_n)) = \prod_{i=1}^{k} p(\tau_m^{(i)}). \tag{13}$$

Hence, we use decomposed preferences as the learning objective, for VPD, its NLL loss for decomposed preferences as

$$\log p(\phi(\tau_n)) = \sum_{i=1}^{k} \log p(\tau_m^{(i)}). \tag{14}$$

This demonstrates that the distillation loss for decomposed preferences on a large batch of responses is equivalent to the sum of losses over multiple smaller batches of responses. A similar decomposition also applies to PPD, and the proof can be found in Appendix A. Based on this insight, we adopt the *Iterative Distillation Process*, as illustrated in Figure 3. We decompose the full distillation process over n responses into k iterations of distillation over m responses each. In this process, the complexity of modeling the preference distribution is reduced from $O(n!)$ to $O(k \cdot m!)$, thereby decreasing the computational cost of training.

Model Families	Method	Alpaca-Eval 2.0		Arena-Hard	MT-Bench	GSM8K
		LC (%)	WR (%)	WR (%)	Score (1∼10)	Acc. (%)
GEMMA-2	Teacher (9B)	55.27	42.50	61.16	6.99	87.41
	Student (2B)	39.51	41.99	37.55	6.70	51.63
	Standard KD	41.67 (↑2.2)	45.24 (↑3.3)	52.36 (↑14.8)	6.78 (↑0.1)	54.37 (↑2.7)
	SeqKD	42.91 (↑3.4)	46.44 (↑4.4)	54.87 (↑17.3)	6.88 (↑0.2)	55.72 (↑4.1)
	MiniLLM	42.97 (↑3.5)	48.32 (↑6.3)	55.75 (↑18.2)	6.88 (↑0.2)	55.26 (↑3.6)
	DPO	43.77 (↑4.3)	54.02 (↑12.0)	57.43 (↑19.9)	6.87 (↑0.2)	57.07 (↑5.4)
	SimPO	44.94 (↑5.4)	54.16 (↑12.2)	58.64 (↑21.1)	6.91 (↑0.2)	57.24 (↑5.6)
	PRO	45.87 (↑6.4)	56.48 (↑14.5)	58.95 (↑21.4)	6.96 (↑0.3)	58.83 (↑7.2)
	PAD w/ $\mathcal{L}_{\text{VPD}}$	46.13 (↑6.6)	57.94 (↑16.0)	59.07 (↑21.5)	6.93 (↑0.2)	59.06 (↑7.4)
	PAD w/ $\mathcal{L}_{\text{PPD}}$	**49.62** (↑10.1)	**59.50** (↑17.5)	**60.00** (↑22.4)	**7.02** (↑0.3)	**59.29** (↑7.7)
LLAMA-3	Teacher (8B)	37.01	38.93	52.66	7.00	84.00
	Student (3B)	27.82	29.02	31.70	6.42	57.09
	Standard KD	29.11 (↑1.3)	29.60 (↑0.6)	41.68 (↑10.0)	6.49 (↑0.1)	59.15 (↑2.1)
	SeqKD	29.48 (↑1.7)	30.04 (↑1.0)	42.52 (↑10.8)	6.53 (↑0.1)	60.94 (↑3.8)
	MiniLLM	30.05 (↑2.2)	30.38 (↑1.4)	42.21 (↑10.5)	6.67 (↑0.3)	60.35 (↑3.3)
	DPO	31.42 (↑3.6)	32.01 (↑3.0)	44.71 (↑13.0)	6.62 (↑0.2)	61.63 (↑4.5)
	SimPO	32.74 (↑4.9)	32.46 (↑3.4)	44.85 (↑13.2)	6.73 (↑0.3)	61.22 (↑4.1)
	PRO	32.11 (↑4.3)	32.23 (↑3.2)	45.09 (↑13.4)	6.71 (↑0.3)	61.47 (↑4.4)
	PAD w/ $\mathcal{L}_{\text{VPD}}$	32.71 (↑4.9)	32.34 (↑3.3)	45.23 (↑13.5)	6.77 (↑0.3)	61.35 (↑4.3)
	PAD w/ $\mathcal{L}_{\text{PPD}}$	**33.61** (↑5.8)	**32.55** (↑3.5)	**46.73** (↑15.0)	**6.84** (↑0.4)	**62.24** (↑5.1)

Table 1: Main results with the Gemma-2 and LLaMA-3 Models.

5 Experiment

5.1 Setup

Models We evaluate two model families in our main experiments: 1) GEMMA-2 Models[3] (Riviere et al., 2024) include GEMMA-2-9B-IT as teacher and GEMMA-2-2B-IT as the student, and 2) LLAMA-3 Models[4] (Grattafiori et al., 2024) includes LLAMA-3.1-8B-INSTRUCT as teacher and LLAMA-3.2-3B-INSTRUCT as student.

Training We construct the training data from ULTRAFEEDBACK[5] (Cui et al., 2023), which comprises around 60k preference data. This dataset covers a broad range of real user prompts, spanning tasks such as mathematical reasoning and open-ended writing. We filter out samples that exceed the context length of the models. We set the number of sampled responses n to 4. To mitigate the reward bias of the teacher model, we set the reward calibration ratio α to 0.8. By default, our training epoch is 1. Detailed experimental settings can be found in Appendix B.1.

Evaluation We evaluate our model on the following four benchmarks: AlpacaEval 2.0 (Li et al., 2023), MT-Bench (Zheng et al., 2023), Arena-Hard (Li et al., 2024), and GSM8K (Cobbe et al., 2021). These benchmarks assess the model's versatile conversational capabilities across various queries and have been widely adopted by the community. For AlpacaEval, we provide both the raw win rate (WR) and the length-controlled win rate (LC) against the reference model. The LC metric is specifically designed to be robust against model verbosity. For Arena-Hard, we report the win rate (WR)[6]. For MT-Bench, we report the average MT-Bench score evaluated by GPT-4 Turbo. Detailed evaluation settings can be found in the Appendix B.2

Baselines We compare PAD with two types of baselines: 1) Traditional Knowledge Distillation, which aims to learn the teacher's distribution at the logits level, including **Standard KD** (Hinton et al., 2015), **SeqKD** (Kim and Rush, 2016), and **MiniLLM** (Gu et al., 2024); 2) Preference Knowledge Distillation, which aims to transfer the teacher's preference knowledge to the student model. Under the "Teacher-as-Annotator" paradigm, we choose **DPO** (Tunstall et al., 2024), **SimPO** (Meng et al., 2024), and **PRO** (Song et al., 2024) as baselines. A detailed description of these baselines can be found in Appendix B.3.

[3] https://ai.google.dev/gemma

[4] https://ai.meta.com/blog/meta-llama-3/

[5] https://huggingface.co/datasets/argilla/ultrafeedback-binarized-preferences-cleaned

[6] Please note that for AlpacaEval 2.0 and Arena-Hard, we employ LLAMA-3.1-70B-INSTRUCT as the judge model, which achieved capabilities comparable to GPT-4 Turbo on the judge test of AlpacaEval while being more cost-effective and faster.

5.2 Main Result

Table 1 presents the main experimental results across multiple benchmarks, including Alpaca-Eval 2.0, Arena-Hard, MT-Bench, and GSM8K. The key finding is that the student models trained by PAD consistently outperform its initial counterpart and existing approaches, achieving over 20% improvement on AlpacaEval 2.0 and Arena-Hard, demonstrating that PAD is able to more precisely capture the teacher's preferences, thereby better aligning with human values.

When comparing PAD with traditional knowledge distillation (KD) and preference distillation methods, PAD demonstrates a clear edge. Traditional KD methods, such as Standard KD and SeqKD, show limited improvement over the initial student model, with performance improvements ranging from 1-3% in Alpaca-Eval 2.0 LC scores. In contrast, distillation methods like DPO and SimPO show more significant improvements, particularly in aligning better with human preferences, as observed in the Alpaca-Eval 2.0 and Arena-Hard benchmarks, which is consistent with the findings of Tunstall et al. (2024).

PAD w/ $\mathcal{L}_{\text{PPD}}$ consistently outperforms PAD w/ $\mathcal{L}_{\text{VPD}}$, with notable gains across benchmarks such as Alpaca-Eval 2.0 LC (49.62 vs. 46.13) and MT-Bench (7.02 vs. 6.93). In particular, for the GEMMA-2 model family, the student trained with $\mathcal{L}_{\text{PPD}}$ even slightly surpasses the teacher model in MT-Bench, achieving a score of 7.02 compared to the teacher's 6.99. The key advantage of PPD over VPD lies in the preference modeling strategy. Instead of just giving a simple preference ranking, modeling the full preference distribution provides more nuanced supervisory signals. This enhancement is crucial, as it better captures subtle human preference, a factor often overlooked in existing distillation methods.

5.3 Analysis

We analyze the impact of our proposed Preference Decomposing Strategy and Reward Calibration. To explore the generalization capability of PAD, we also analyze the performance when the teacher and student come from different model families. More detailed analyses can be found in Appendix C.

Effect of Preference Decomposing Strategy Based on the preference decomposing strategy, we investigated the impact of the iterative distillation process on performance and training time. Table 2 illustrates the effects of different numbers of iterations and sample sizes. When the sample size is 4, decomposing the sampling process into two iterative steps does not reduce training time, because the complexity of modeling the distribution with fewer samples is low and thus negligible. However, when the sample size increases to 8, adopting a two-iteration decomposition shortens the time by 12%, indicating that as the sample size increases, the preference decomposing strategy becomes more advantageous in accelerating training. Moreover, decomposing the sampling into multiple iterative steps does not lead to a significant performance drop, demonstrating that this strategy can maintain stable performance while improving efficiency.

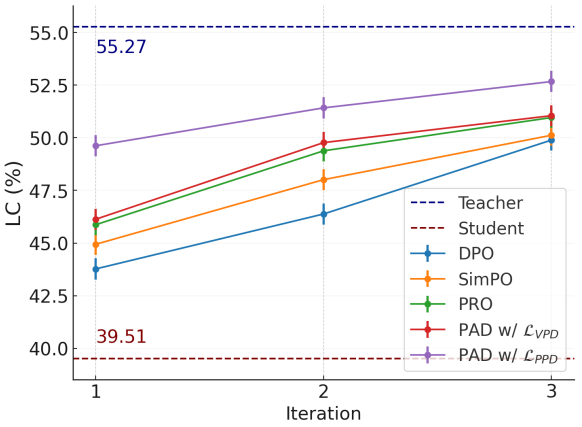


Figure 4: Alpaca-Eval 2 LC with different iterations.

Iteration / Sample Size	Alpaca-Eval 2 LC (%)	GPU Hours A800
1 / 4	49.42	12.32
2 / 2	48.94	12.35
1 / 8	50.57	31.51
2 / 4	51.42	27.98

Table 2: Impact of different number of iterations and sample sizes on performance and training time.

We further explored the impact of the iterative distillation process as a continuous learning method. In Figure 4, we compared three high-performing baseline methods: DPO, SimPO, and PRO. The results show that the iterative distillation process enhances performance across all methods, with our PAD achieving the best results, highlighting its effectiveness.

Influence of Reward Calibration Ratio We investigate the effect of the calibration ratio α in our PAD (Figure 5). Without reward calibration ($\alpha = 0$), the performance improvement of the distilled student model is marginal, possibly due to mis-calibration of sequence likelihood. Increasing α improves performance, with the best results at

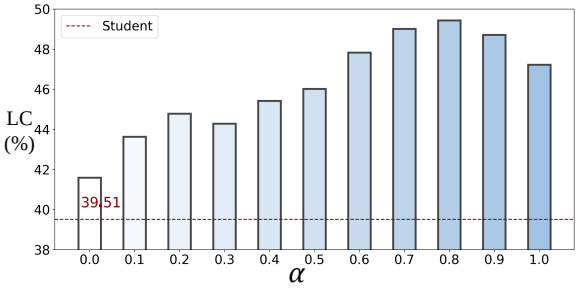


Figure 5: Alpaca-Eval 2 LC Win Rate with different α

Method	Alpaca-Eval	Arena-Hard
	LC (%)	WR (%)
Student	27.82	31.70
DPO	49.42(↑21.6)	56.72(↑25.0)
SimPO	49.78(↑22.0)	56.69(↑25.0)
PRO	50.18(↑22.4)	58.23(↑26.5)
PAD w/ $\mathcal{L}_{\text{VPD}}$	50.45(↑22.6)	58.47(↑26.8)
PAD w/ $\mathcal{L}_{\text{PPD}}$	**51.96**(↑24.1)	**59.96**(↑28.3)

Table 3: Heterogeneous Distillation Study. We use GEMMA-2-9B-IT as the teacher and LLAMA-3.2-3B-INSTRUCT as the student.

$\alpha = 0.8$. However, further increases in α offer diminishing returns and may slightly decrease performance. For lower-performance requirements, using $\alpha = 1$ is a viable option, as it avoids the computational cost of calculating the teacher's log-likelihood.

Heterogeneous Study In previous experiments, both the teacher and the student models belonged to the same model family, indicating that they share the same vocabulary and have similar architectures. To verify the generalization capability of our PAD, we conducted experiments where the teacher and student models come from different model families. As shown in Table 3, our method consistently outperforms other approaches, achieving a 24.1 improvement in LC on Alpaca-Eval and a 28.3 improvement in WR on Arena-Hard. These results demonstrate its generalization ability.

6 Related Work

Traditional Knowledge Distillation Knowledge distillation (KD), introduced by Hinton et al. (2015), primarily aims at model compression by training a smaller student model to mimic the output behavior of a larger teacher model (Kim and Rush, 2016; Liang et al., 2021; Zhang et al., 2023; Gu et al., 2024; Agarwal et al., 2024). Kim and Rush (2016) extended KD to machine translation by training students on sequences generated by teachers in order to imitate teacher behavior. More recently, Gu et al. (2024) advanced KD using reverse KL divergence on student -generated sequence to mitigate exposure bias, improving student model performance. A key feature of these methods is that distillation is performed over the shared vocabulary of both teacher and student models. Our PAD eliminates this limitation, enabling effective distillation with different vocabularies.

Preference Knowledge Distillation Motivated by the observation that large models have achieved a high degree of alignment with human values and preferences, many efforts focus on distilling preference knowledge from large models to smaller ones (Bai et al., 2022; Cui et al., 2023; Lee et al., 2024; Yuan et al., 2024; Tunstall et al., 2024; Yang et al., 2024). Bai et al. (2022) first introduced this concept, also known as Reinforcement Learning from AI Feedback (RLAIF), where teacher models annotate response pairs from the student to create a preference dataset for training a reward model. Tunstall et al. (2024) further utilized teacher-annotated preferences with Direct Preference Optimization (DPO) (Rafailov et al., 2023), streamlining the training of student models. These approaches follow the "Teacher-as-Annotator" paradigm. The annotated preference datasets generated through this paradigm can be directly employed with methods such as DPO, SimPO (Meng et al., 2024), and PRO (Song et al., 2024), enabling preference optimization of student models. However, a significant limitation of these methods lies in their reliance on unique ranking, which constrains their ability to model nuanced preferences. In contrast, our PAD treats modeling preference knowledge as a distribution over all possible preferences, enabling nuanced alignment for the student and teacher models.

7 Conclusion

In this paper, we introduced the Preference-Aligned Distillation (PAD) framework, in which we model the teacher's preference knowledge as a probability distribution over all possible preferences. This supervisory signal allows the student to learn the subtle differences between responses. Our experiments on the GEMMA-2 and LLAMA-3 model families demonstrated that PAD outperforms traditional knowledge distillation and existing preference distillation methods across four benchmarks, showcasing its emergent capability of learning in-depth human preferences.

Limitations

Our research has several limitations. Firstly, the generalization capability is insufficient as we have not conducted experiments on larger-scale teacher and student models, primarily due to limited computational resources. Secondly, sampling multiple responses consumes more computational overhead. However, because SLMs have relatively smaller parameter sizes, this overhead remains comparatively modest. Thirdly, our method requires token-level probabilities, which are unavailable in some black-box models.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*.

Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Direct preference optimization with an offset. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9954–9972, Bangkok, Thailand. Association for Computational Linguistics.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback. *Preprint*, arXiv:2212.08073.

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *Preprint*, arXiv:2407.21787.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129–136, New York, NY, USA. Association for Computing Machinery.

Alex James Chan and Mihaela van der Schaar. 2021. Scalable bayesian inverse reinforcement learning. In *International Conference on Learning Representations*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *Preprint*, arXiv:2310.01377.

Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, and Lexi Walker et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *Preprint*, arXiv:2403.05530.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, and Llama Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2024. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*.

Yongchang Hao, Yuxin Liu, and Lili Mou. 2022. Teacher forcing recovers reward functions for text generation. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2024. RLAIF: Scaling reinforcement learning from human feedback with AI feedback.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. 2024. From live data to high-quality benchmarks: The arena-hard pipeline.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. 2021. Mixkd: Towards efficient distillation of large-scale language models. *Preprint*, arXiv:2011.00593.

R.D. Luce. 2012. *Individual Choice Behavior: A Theoretical Analysis*. Dover Books on Mathematics. Dover Publications.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

R. L. Plackett. 1975. The analysis of permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 24(2):193–202.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. 2023a. Robots that ask for help: Uncertainty alignment for large language model planners. In *Proceedings of the Conference on Robot Learning (CoRL)*.

Jie Ren, Yao Zhao, Tu Vu, Peter J. Liu, and Balaji Lakshminarayanan. 2023b. Self-evaluation improves selective generation in large language models. In *Proceedings on "I Can't Believe It's Not Better: Failure Modes in the Age of Foundation Models" at NeurIPS 2023 Workshops*, volume 239 of *Proceedings of Machine Learning Research*, pages 49–64. PMLR.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, and Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18990–18998.

Lewis Tunstall, Edward Emanuel Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M Rush, and Thomas Wolf. 2024. Zephyr: Direct distillation of LM alignment. In *First Conference on Language Modeling*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yufei Wang, Zhanyi Sun, Jesse Zhang, Zhou Xian, Erdem Biyik, David Held, and Zackory Erickson. 2024. RL-VLM-f: Reinforcement learning from vision language foundation model feedback. In *Forty-first International Conference on Machine Learning*.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024. RLCD: Reinforcement learning from contrastive distillation for LM alignment. In *The Twelfth International Conference on Learning Representations*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Rongzhi Zhang, Jiaming Shen, Tianqi Liu, Jialu Liu, Michael Bendersky, Marc Najork, and Chao Zhang. 2023. Do not blindly imitate the teacher: Using perturbed loss for knowledge distillation. *Preprint*, arXiv:2305.05010.

Yao Zhao, Mikhail Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J Liu. 2023. Calibrating sequence likelihood improves conditional language generation. In *The Eleventh International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. 2008. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438.

hyperparameter	Value	Searching Space
β	10	$[1, 2, 5, 8, 10]$
batch size	128	$[32, 64, 128]$
warmup ratio	0.1	$[0.05, 0.1]$

Table 4: The hyperparameter values in PAD training.

Model	Human Agreement
GPT4	69.17
LLAMA-3.1-70B-INSTRUCT	69.10
GPT4-Turbo	68.09
LLAMA-3-70B-INSTRUCT	67.53
QWEN2.5-72B-INSTRUCT	67.51
Humans	65.66

Table 5: Leaderboard of judge models in AlpacaEval.

A Decomposing Probabilistic Preference Distillation

Substituting Eq. 13 into the KLD:

$$\sum_{\tau_n}\left(\prod_{i=1}^{k} p_{\pi^{\text{tch}}}(\tau_m^{(i)})\right)\log\left(\frac{\prod_{i=1}^{k} p_{\pi^{\text{tch}}}(\tau_m^{(i)})}{\prod_{i=1}^{k} p_{\pi^{\text{mix}}}(\tau_m^{(i)})}\right)$$
$$=\sum_{\tau_n}\left(\prod_{i=1}^{k} p_{\pi^{\text{tch}}}(\tau_m^{(i)})\right)\left(\sum_{i=1}^{k}\log\frac{p_{\pi^{\text{tch}}}(\tau_m^{(i)})}{p_{\pi^{\text{mix}}}(\tau_m^{(i)})}\right)$$

interchange summations,

$$=\sum_{i=1}^{k}\sum_{\tau_n}\left(\prod_{j=1}^{k} p_{\pi^{\text{tch}}}(\tau_m^{(j)})\right)\log\frac{p_{\pi^{\text{tch}}}(\tau_m^{(i)})}{p_{\pi^{\text{mix}}}(\tau_m^{(i)})}$$

notice that for a fixed i, the logarithm term only depends on $\tau_m^{(i)}$, and the product can be separated,

$$=\sum_{i=1}^{k}\left(\sum_{\tau_m^{(i)}} p_{\pi^{\text{tch}}}(\tau_m^{(i)})\log\frac{p_{\pi^{\text{tch}}}(\tau_m^{(i)})}{p_{\pi^{\text{mix}}}(\tau_m^{(i)})}\right)\prod_{j\neq i}\left(\sum_{\tau_m^{(j)}} p_{\pi^{\text{tch}}}(\tau_m^{(j)})\right)$$

based on the independence assumption of sub-preferences, $\sum_{\tau_m^{(j)}} p_{\pi^{\text{tch}}}(\tau_m^{(j)}) = 1$ for each j,

$$=\sum_{i=1}^{k}\sum_{\tau_m^{(i)}} p_{\pi^{\text{tch}}}(\tau_m^{(i)})\log\frac{p_{\pi^{\text{tch}}}(\tau_m^{(i)})}{p_{\pi^{\text{mix}}}(\tau_m^{(i)})}$$

Therefore, the KLD can be decomposed as:

$$\mathcal{D}_{\text{KL}}(p_{\pi^{\text{tch}}}(\tau_n)\|p_{\pi^{\text{mix}}}(\tau_n)) = \sum_{i=1}^{k}\mathcal{D}_{\text{KL}}(p_{\pi^{\text{tch}}}(\tau_m^{(i)})\|p_{\pi^{\text{mix}}}(\tau_m^{(i)}))$$

The JSD Loss (Eq. 12) used in PPD is the average of two KLDs in different directions, making JSD also decomposable.

B Implementation Details

B.1 Training

We individually search the learning rates for different model families in the range of $[3e-7, 5e-7, 8e-7, 1e-6, 1e-5]$. As a result, the learning rate for GEMMA-2 Models is $8e-7$ and for LLAMA-3 Models is $1e-6$. Table 4 shows other hyperparameters for training. All the training experiments in this paper were conducted on 2×A800 GPUs based on the TRL repo[7].

B.2 Evaluation

Data Statistics AlpacaEval 2 consists of 805 questions from five datasets, MT-Bench includes 80 questions across eight categories, and the recently released Arena-Hard is an enhanced version of MT-Bench, comprising 500 challenging questions. Since the training data, ultrafeedback, includes some mathematical reasoning problems, we additionally incorporate the GSM8K test set, which contains approximately 1,300 questions, to evaluate the model's mathematical abilities.

Judge Models For AlpacaEval 2.0 and Arena-Hard, we employ LLAMA-3.1-70B-INSTRUCT as the judge model. For MT-Bench, we employ GPT-4 Turbo as the judge model. For GSM8K, we report accuracy on the test set. Table 5 presents the evaluation capability test[8] of these judge models on AlpacaEval. We can see that LLAMA-3.1-70B-INSTRUCT has evaluation capabilities comparable to GPT4-Turbo.

B.3 Baselines

For traditional knowledge distillation, we consider three baselines: 1) **Standard KD** (Hinton et al., 2015): Fine-tunes the student model using the teacher model's logits distribution as a supervision signal, applied to golden responses. 2) **SeqKD** (Kim and Rush, 2016): Directly fine-tunes the student model with cross-entropy loss using responses generated by the teacher model. 3) **MiniLLM** (Gu et al., 2024): Employs the teacher model's logits

[7] https://github.com/huggingface/trl/tree/main

[8] https://github.com/tatsu-lab/alpaca_eval/tree/main/src/alpaca_eval/evaluators_configs

distribution as supervision signal while fine-tuning the student model on its own generated responses.

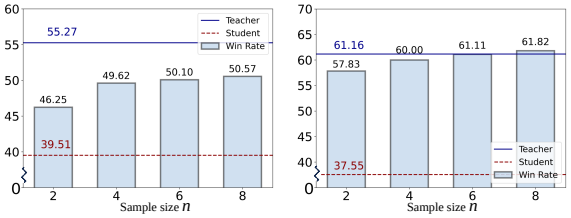
For preference knowledge distillation, under the "Teacher-as-Annotator" paradigm, we employ three offline preference optimization methods as baselines: 1) **DPO** (Rafailov et al., 2023; Tunstall et al., 2024): Treats the student model as a reward model, fine-tuning it based on a reward function derived from a reference model. 2) **SimPO** (Meng et al., 2024): Operates similarly to DPO but uses average log-likelihood as the optimization objective. 3) **RPO** (Song et al., 2024): Extends the above approaches by optimizing with listwise preference information. For a fair comparison, we also use responses sampled from the student model for these baselines. We use the MCQ selection probability introduced in Section 4.2 as the score to rank the responses. For the pairwise preference optimization methods DPO and SimPO, we select the responses with the maximum and minimum rewards to form preference pairs. For the listwise preference optimization method PRO, we directly sort the scores to form the preference ranking. Please kindly note that constructing preference pairs using the maximum and minimum scores of responses is a common practice (Cui et al., 2023; Meng et al., 2024). Moreover, our preliminary experiments indicate that splitting the entire listwise response data into multiple pairwise data and training with DPO/SimPO does not yield significant performance improvements.

C Additional Experiments and Analyses

Method	Alpaca-Eval	Arena-Hard
	LC (%)	WR (%)
Teacher	56.89	76.09
Student	39.51	37.55
DPO	51.93	65.42
SimPO	52.36	66.36
PRO	52.45	68.01
PAD w/ $\mathcal{L}_{\text{VPD}}$	53.32	67.38
PAD w/ $\mathcal{L}_{\text{PPD}}$	55.96	69.90

Table 6: Scaling-up Study.

Scaling Up We use GEMMA-2-27B-IT as the teacher and GEMMA-2-2B-IT as the student. The overall performance is shown in Table 6. When employing larger-scale teacher models, our PAD consistently and significantly enhances the ability of small models to align with human preferences. Compared to the main result (§5.2), we observe



(a) Alpaca-Eval 2.0 LC Win Rate (%) (b) Arena-Hard Win Rate (%)

Figure 6: Win Rate with different sample size n.

that when using a more capable teacher, the student model achieves greater performance improvements, indicating that the performance gap between the teacher and student is a key factor in determining the extent of the student's enhancement.

Effect of Sample Size We investigate the impact of the number of sampled responses on PPD, and the results can be seen in Figure 6. We observe that as the number of sample size n increases, the performance of the student model improves accordingly. This indicates that obtaining more feedback knowledge through extensive sampling from the text generation space facilitates better alignment of the student model with the teacher's preferences.