Spread Preference Annotation: Direct Preference Judgment for Efficient LLM Alignment

Dongyoung Kim[1], Kimin Kim[1], Jinwoo Shin[1], Jaehyung Kim[2]

[1]KAIST AI, [2]Yonsei University

Summary

❒ **Why? Human preference labeling is costly**. Existing methods depend on large aligned LLMs or reward models, which are both data-hungry and noise-prone

❒ **How?** SPA leverages **logits-based preference judgment and self-generated data**, combined with noise-aware refinement—**no additional training models** required

❒ **What?** SPA achieves **+16.4%** AlpacaEval win rate using only **3.3%** of gold labels. Outperforms strong baselines and even works with zero human-labeled data

Introduction

❒ Challenge: Aligning LLMs with Human Preferences

➢ Aligning LLMs with user intent is crucial but requires costly, large-scale human preference data.

❒ Limitations of Existing Approaches

➢ LLM-as-judge and reward models either require strong base models or large labeled datasets

➢ LLM-as-judge is AI feedback and differs from actual human preferences

❒ Our Solution: Spread Preference Annotation (SPA)

➢ Leveraging human prior knowledge within the small (seed) data a and progressively improving the alignment of LLM

➢ Utilize implicit reward from LLM to explicitly extract the model's inherent preference

➢ Improving learning performance by automatically processing noise during the learning process

$$p_\theta(y_w \succ y_l|x) = \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)$$

Method: SPA

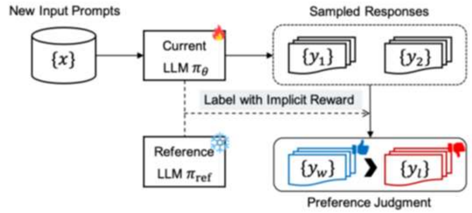
❒ Step 0: Initialization

➢ Train the initial model using DPO on seed dataset

$$\mathcal{L}_{\text{DPO}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[-\log p_\theta(y_w \succ y_l|x)\right].$$

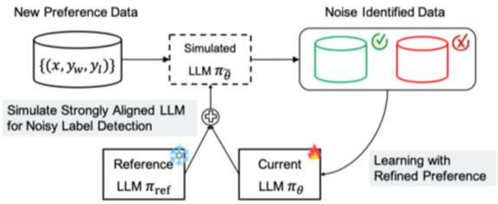
❒ Step 1: Preference Data Generation

➢ Sample new responses for prompts and use the model's logits to infer preference labels via implicit reward comparison



$$p_{i-1}(y_1 \succ y_2|x) = \sigma\left(\beta \log \frac{\pi_{i-1}(y_1|x)}{\pi_{\text{init}}(y_1|x)} - \beta \log \frac{\pi_{i-1}(y_2|x)}{\pi_{\text{init}}(y_2|x)}\right)$$

❒ Step 2: Self-Refined Preference Learning



➢ If the internal preference of the model being trained does not exceed a certain threshold, it is treated as noise

$$z_\theta = 1 \text{ if } p_\theta(y_w \succ y_l|x) < \tau \text{ else } z_\theta = 0,$$

➢ Noise labels are given low weight in preference learning

$$\mathcal{L}_{\text{rf}}(\pi_\theta) = \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_i}\left[-\left((1-\alpha * z_\theta)\log p_\theta(y_w \succ y_l|x) + \alpha * z_\theta \log p_\theta(y_l \succ y_w|x)\right)\right],$$

➢ More strong aligned logit is obtained from linear combination of the logits of π_θ and π_{ref}, and noise is detected based on this

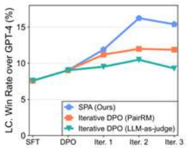
$$h_{\tilde{\theta}}(x, y_{1:t-1}) = (1+\lambda) * h_\theta(x, y_{1:t-1}) - \lambda * h_{\text{ref}}(x, y_{1:t-1}),$$

Experiment Result

❒ Results with 3.3% Gold Labels

➢ We train Mistral-7b-01v on Ultrafeedback datasets

Models	Gold Label (%)	AlpacaEval 2.0 Len-control. Win Rate (%)	AlpacaEval 2.0 Win Rate vs. GPT-4 (%)	MT-Bench Avg. Score (0-10)
Mistral-7B-v0.1	-	0.17	0.50	3.25
Zephyr-7b-β	100	11.75	10.03	6.87
SFT	-	7.58	4.72	6.34
DPO	3.3	9.03	7.68	6.81
SPA (Ours)	3.3	**15.39**	**21.13**	**6.94**



❒ Effect of Seed Size on Performance (AlpacaEval 2.0)

Methods	Used Ground-truth Preference Data 0.8%	1.7%	3.3%	10%
DPO: LC Win Rate (%)	7.85	7.68	9.03	11.37
DPO: Win Rate (%)	5.53	5.49	7.68	9.32
SPA: LC Win Rate (%)	10.36	12.36	16.23	18.52
SPA: Win Rate (%)	11.34	13.72	19.94	23.79

Limitation & Future Work

❒ Limitation

➢ There is a large length bias, may be many other biases.
➢ Vulnerable to error accumulation

❒ Enhanced noise handling

➢ A methodology that separates and reduces bias at the model and dataset level, such as co-teaching, would be effective.
➢ When using implicit rewards, such as offline RL methodologies, strong regularization can be applied.

❒ Research advances using implicit reward

➢ Methodologies that utilize implicit rewards have shown experimentally superior performance in various fields.

- Self Improving: Bootstrapping Language Models with DPO Implicit Rewards (ICLR)
- Process Reward Modeling :Process Reinforcement through Implicit Rewards