ARF-RLHF: Adaptive Reward-Following for RLHF through Emotion-Driven Self-Supervision and Trace-Biased Dynamic Optimization

Anonymous ACL submission

Abstract

Recent advances in reinforcement learning from human feedback (RLHF) and autoregressive transformers have driven the evolution of large language models such as GPT-4.0, DeepSeek R1, and Llama 3.3, enabling richer and more personalized responses. However, prevailing RLHF paradigms—from Proximal Policy Optimization (PPO) to Direct Preference Optimization (DPO)—still rely on binary-preference labels. While this approach reduces immediate annotation costs, it demands extensive human labeling and captures only coarse, group-level tastes. It suffers from high annotation overhead and limited adaptability to individual users. To address these limitations, we introduce Adaptive Reward-Following (ARF), a self-assessment framework that converts free-form user feedback into continuous preference signals using a high-precision satisfaction scorer (70% accuracy on GoEmotions, Sentiment140, and DailyDialog). We further refine and debias these signals through data augmentations—synonym replacement, trace truncation, and score bias annotation—and use a Dynamic Adapter Preference Tracker to model evolving user tendencies in real time. Building on these components, our novel Trace Bias (TB) fine-tuning algorithm optimizes continuous reward trajectories rather than binary labels. Experiments on Qwen-2/2.5, Gemma-2, and Llama-3.2 across four preference domains show that ARF outperforms PPO by 3.3% and DPO by 7.6%, while maintaining alignment with RLHF objectives. ARF delivers a scalable, personalized, and cost-effective paradigm for next-generation RLHF in large language models.

1 Introduction

The rapid evolution of Large Language Models (LLMs) has dramatically improved performance across a wide variety of tasks. As these models become more robust and generally capable, the focus has shifted from factual accuracy toward deeper personalization and alignment with individual user intent. Reinforcement Learning from Human Feedback (RLHF) has emerged as the de facto fine-tuning strategy—powering systems like GPT-4 (OpenAI et al., 2024), DeepSeek-R1 (DeepSeek-AI et al., 2025), and Llama-3 (Grattafiori et al., 2024). Yet, prevailing RLHF pipelines (e.g., PPO (Schulman et al., 2017b), DPO (Rafailov et al., 2024)) hinge on massive, binary-preference annotations, often involving hundreds or thousands of crowdworkers. This approach incurs substantial human cost and only captures coarse, group-level tendencies rather than the nuanced tastes of each user.

To mitigate annotation overhead and broaden participation, prior work has explored variants such as RLAIF (Lee et al., 2024b) and crowd-sourced preference distillation (Zhang et al., 2024), which convert coarse feedback into richer signals and improve sample efficiency. While these methods reduce labeling burden and enhance model generalization, they still depend on externally sourced human annotations and manual prompt engineering. Crucially, without fundamentally rethinking how feedback is collected, they cannot track true individual preference trajectories—doomed to repeat the same group-bias limitations.

In this paper, we introduce Adaptive Reward-Following (ARF), a self-supervised RLHF framework designed to minimize manual labeling while capturing fine-grained, per-user preferences. As illustrated in Figure 1, our pipeline consists of four stages: (1) Automatically collecting satisfaction QA data from user feedback; (2) Augmenting and storing samples via synonym substitution, truncation, and preference-biased reweighting to build a diversified reward corpus; (3) Training and continuously updating a lightweight ARF scorer with soft labels to predict satisfaction scores; (4) Fine-tuning the LLM using the TraceBias algorithm, which

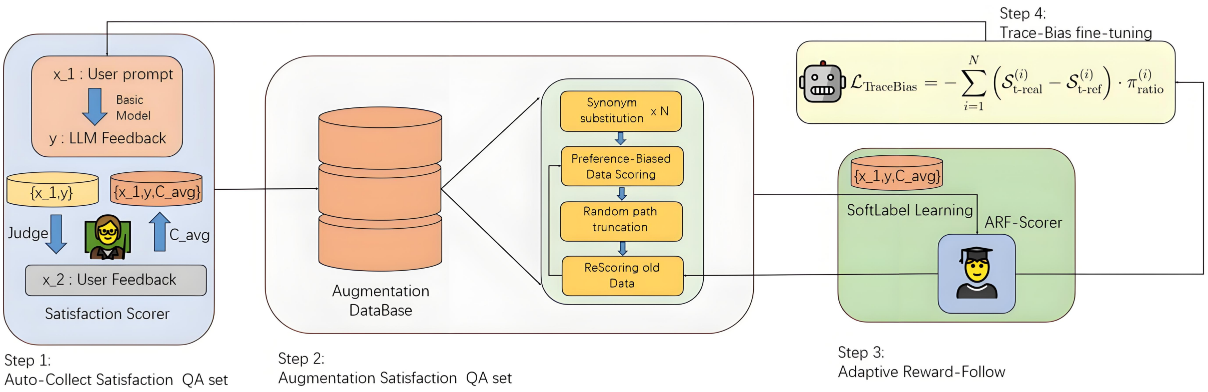

Figure 1: Illustrates the overall workflow of our framework. We begin by automatically collecting satisfaction QA data based on user feedback (Step 1). These samples are then stored and augmented through synonym substitution, truncation, and reweighting to form a diversified reward corpus (Step 2). The ARF scorer is trained with soft labels to predict satisfaction scores and is continuously updated (Step 3). Finally, the TraceBias algorithm leverages ARF-generated rewards to fine-tune the LLM (Step 4), completing a fully self-supervised RLHF pipeline.

directly optimizes on ARF-generated soft reward trajectories. This fully self-supervised workflow not only eliminates the need for binary comparisons and significantly reduces annotation costs, but also makes precise modeling of individual user preferences possible.

Empirical evaluations on Qwen-2/2.5 (Team et al., 2024; Qwen et al., 2025), Gemma-2 (Team et al., 2024), and Llama-3.2 (Grattafiori et al., 2024) across four diverse preference domains show that ARF outperforms PPO by 3.3% and DPO by 7.6%, while preserving theoretical alignment with established RLHF objectives. These results demonstrate that ARF provides a scalable, cost-effective route to truly personalized LLM fine-tuning.

2 Background

2.1 The Core Pipeline of RLHF for LLMs

Despite design variations, The Bradley-Terry(BT) model and policy gradient optimization form the foundation of RLHF training. While implementations vary (e.g., PPO (Schulman et al., 2017b) and DPO (Rafailov et al., 2024)), all methods share these core components.

Supervised Fine-Tuning (SFT): The pipeline begins with SFT on large-scale corpora to instill general knowledge in LLMs. For downstream applications, domain-specific datasets are used for further fine-tuning. While this results in high task-specific accuracy, such models often lack adaptability and personalization in real-world interactive settings.

Preference Data Collection: To incorporate human feedback, a preference dataset is constructed using paired outputs (e.g., `(prompt, answer1, answer2)`). A Bradley-Terry (BT) model (BRADLEY, 1955) is commonly used to estimate the preference probability between responses $y_w \succ y_l$ as:

$$\mathcal{P}(y_w \succ y_l|x) = \frac{\exp(\mathcal{R}(x, y_w))}{\exp(\mathcal{R}(x, y_w)) + \exp(\mathcal{R}(x, y_l))}, \tag{1}$$

where $\mathcal{R}(x, y)$ denotes a learned reward function.

Reinforcement Learning with Human Feedback: RLHF methods rely primarily on: (1) reward modeling from preferences, and (2) policy gradient(PG) optimization. The policy is trained to maximize expected return:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t \right], \tag{2}$$

where $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ is the discounted return. Advantage-based methods refine this further:

$$\nabla_\theta J(\theta) = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot A^\pi(s_t, a_t) \right], \tag{3}$$

with the advantage function defined as:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t). \tag{4}$$

The multi preference-based dataset and PG optimization constitutes the foundation of RLHF training, often enhanced by optimization techniques such as Trust Region Policy Optimization (TRPO) (Schulman et al., 2017a), Rank Responses to Align Language Models with Human Feedback (RRHF) (Yuan et al., 2023), and Reinforcement Learning from Human Feedback with AI Feedback(RLAIF) (Lee et al., 2024a).

2.2 Preference-Based Optimization has a Constant Foundation

While RLHF methods continue to evolve, most remain grounded in the BT preference modeling and policy gradient framework defined in Section 2.1. To demonstrate this, we analyze the two dominant approaches: PPO and DPO.

PPO-Based Optimization: Building on Equation 2.1, PPO first trains a reward model via Bradley-Terry ranking loss:

$$\mathcal{L}_{\text{R}} = -\mathbb{E}_{(x,y_w,y_l)} \log \sigma(\mathcal{R}(x, y_w) - \mathcal{R}(x, y_l)), \tag{5}$$

then applies clipped policy gradients to optimize π_θ:

$$\mathcal{L}_{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), \epsilon)\hat{A}_t \right) \right]. \tag{6}$$

DPO as a Reward-Free Alternative: DPO bypasses explicit reward modeling by directly optimizing function's preference margin:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)} \log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \tag{7}$$

where β is a temperature parameter and π_{ref} is a reference policy.

In essence, PPO and DPO are structurally similar—both optimize over preference pairs to align models with human intent. Their main difference lies in whether an explicit reward model is used. However, both remain reliant on human-generated comparisons and operate within the constrained policy optimization framework, which fundamentally limits their autonomy and scalability, We have conducted a more in-depth theoretical analysis in the appendix B.

3 Methodology

Built upon the structural foundation outlined in Section 2.1, ARF-RLHF comprises three core training stages: model initialization, preference scoring, and policy optimization. Model initialization adopts an open-source state-of-the-art LLM. Preference scoring is achieved through a composite scorer that integrates a high-precision satisfaction scorer with a ARF scorer, which continuously adapts over time. These evolving scoring signals directly inform policy optimization, carried out using the TraceBias algorithm. TraceBias performs continuous fine-tuning based on soft reward signals inferred from the scoring system, thus eliminating reliance on binary labels and enabling more nuanced policy updates.

The following subsections elaborate on each component: we begin with the scoring system—the interplay between the static and preference tracker. Next, we describe the score-shift mechanism within the augmentation database, including temporal bias correction and dual-scorer alignment. Finally, we analyze the TraceBias algorithm and its role in optimizing policies within the ARF-RLHF pipeline.

3.1 Adaptive Reward-Following(ARF) Scorer

Recent studies suggest that human communication not only conveys explicit semantic content but also implicitly reflects user satisfaction and willingness to continue the interaction (Chen and Chen, 2016; Shanahan et al., 2006; Henry et al., 2021; Prabhumoye et al., 2017). Building upon this observation, we introduce two complementary scorers: a static satisfaction scorer for collecting quality estimates, and an ARF scorer that periodically updates to capture evolving individual preferences directly from interactions.

3.1.1 Static Satisfaction Scorer

Both the static satisfaction scorer and the ARF scorer are built upon the lightweight RoBERTa-mini (Liu et al., 2019) architecture, balancing low latency with strong semantic understanding.

To enable self-supervised reward modeling, the static scorer predicts the quality of a given (prompt, response) pair based on the user's subsequent reply. Specifically, it takes the user's follow-up message as input and outputs a satisfaction score reflecting the user's sentiment toward the previous system response.

We project the final hidden states of RoBERTa-mini to three sentiment classes: *bad*, *neutral*, and *good*, and aggregate token-level logits to obtain sequence-level satisfaction distribution:

$$\mathcal{C}_3 = \text{Linear}(\mathcal{H}_{\text{Last}}) \tag{8}$$

$$\mathcal{C}_{\text{avg}} = \text{Softmax}\left(\frac{1}{n} \sum_{i=1}^{n} \mathcal{C}_3^{(i)} \right) \tag{9}$$

Here, $\mathcal{H}_{\text{Last}}$ denotes the final hidden states from RoBERTa-mini, and $\mathcal{C}_3 \in \mathbb{R}^{n\times 3}$ represents token-level satisfaction logits, where n is the input sequence length. The final prediction $\mathcal{C}_{\text{avg}} \in \mathbb{R}^3$ sum-

Figure 2: We compare the gradient norm statistics of PPO, using a clip range $\epsilon = 0.2$ as in the original paper (Schulman et al., 2017b) and TraceBias with DAM. DAM exhibits lower variance and more stable gradient magnitudes, suggesting improved training stability and potential for enhanced performance.(V is shown in appendix I)

marizes sequence-level satisfaction through mean pooling followed by softmax normalization. The importance of three sentiment classes structure is further discussed in Discussion 5.2.

These static predictions are collected as soft-labels to train the ARF scorer, which learns to assign reward scores to collected or new (prompt, response) pairs in an offline fashion. The ARF scorer then serves as the reward function in TraceBias, guiding LLM fine-tuning without requiring manual preference annotations.

3.1.2 ARF Scorer

During user interaction, the ARF scorer is periodically fine-tuned to adapt to evolving user preferences. At each training step, it outputs a predicted satisfaction distribution $\hat{\mathcal{C}}$ for LLM feedback. The prediction is supervised using soft labels derived from user feedback or historical interactions.

To accelerate convergence while preserving baseline estimation, the ARF scorer builds on the static satisfaction scorer. The averaged satisfaction vector $\mathcal{C}_{\text{avg}}$, distilled from user follow-up messages, provides soft guidance for supervision. We employ the standard cross-entropy loss for training:

$$\mathcal{L}_{\text{supervised}} = \text{CE}(\hat{\mathcal{C}}, \mathcal{C}_{\text{avg}}) \tag{10}$$

Building on this, to further address overfitting and catastrophic forgetting when real-time data is scarce, we introduce an Experience Replay (ER) mechanism. A sampling ratio ER_{ratio} probabilistically switches training between historical data and current feedback. Specifically:

$$\mathcal{L}_{\text{total}} = \begin{cases} \mathcal{L}_{\text{ER}} = \text{CE}(\hat{\mathcal{C}}, \mathcal{C}_{\text{static}}), & \text{if } p < \text{ER}_{\text{ratio}} \\ \mathcal{L}_{\text{supervised}} = \text{CE}(\hat{\mathcal{C}}, \mathcal{C}_{\text{avg}}), & \text{otherwise} \end{cases}$$

Here, $\mathcal{C}_{\text{static}}$ denotes labels from the static satisfaction dataset (e.g. DailyDialog (Li et al., 2017), GoEmotions (Demszky et al., 2020)), and p is a random sampling variable. This alternating strategy allows the ARF scorer to benefit from both stable historical signals and dynamic user feedback, improving generalization and robustness.

3.2 Augmentation Database

Leveraging the adaptive scoring foundation and aiming to make better use of the limited real user feedback, we construct an Augmentation Database that expands training data via synonym substitution and random trace truncation. To ensure augmented samples reflect evolving user preferences, we introduce a score-shift mechanism that blends static and adaptive sentiment distributions.

3.2.1 Preference-Biased Data Scoring Algorithm

Directly applying the ARF-scorer to evaluate synonym-augmented samples is suboptimal, especially in the early stages when the scorer has not yet adapted to the user's evolving preferences. To address this, we propose a preference-biased data scoring algorithm that considers both the static satisfaction scorer's output $\mathcal{C}_{\text{basic\_avg}}$ and the ARF-scorer's output $\mathcal{C}_{\text{ARF\_avg}}$.

We compute a dynamic weighting coefficient $\mathcal{S}_{\cos}$ based on the cosine similarity between the two score vectors, adjusted via a sigmoid function:

$$\mathcal{S}_{\cos} = \sigma\Big(\big(\text{CosSim}(\mathcal{C}_{\text{ARF\_avg}}, \mathcal{C}_{\text{basic\_avg}}) - 0.5\big) \cdot S_{\text{sig}}\Big) \tag{11}$$

Here, σ denotes the sigmoid function, and S_{sig} is a scaling hyperparameter that controls how sensitive the mixture weight is to score differences (see Appendix D for selection strategy). Since the outputs $\mathcal{C}_{\text{ARF\_avg}}$ and $\mathcal{C}_{\text{basic\_avg}}$ are normalized as in Eq. 8, their cosine similarity falls within $[0, 1]$, which we center around 0 by subtracting 0.5 to achieve a symmetric input range for the sigmoid.

The final score for augmented data $\mathcal{C}_{\text{Aug}}$ is computed as a weighted average of both scores:

$$\mathcal{C}_{\text{Aug}} = \mathcal{C}_{\text{ARF\_avg}} \cdot \mathcal{S}_{\cos} + \mathcal{C}_{\text{basic\_avg}} \cdot (1 - \mathcal{S}_{\cos}) \quad (12)$$

This formula realizes the weighted fusion of static scoring and dynamic preference scoring, ensuring the stability and adaptability of the scoring during the preference evolution process.

3.2.2 Regular Re-Evaluation of Historical Scores

As user preferences naturally evolve over time, previously collected feedback and augmented data may become misaligned with current expectations. However, since the ARF-scorer is continuously updated, using it alone to re-score historical data could result in loss of alignment with past preferences.

To maintain continuity across preference shifts, we propose to regularly update old scores using the same preference-biased data scoring mechanism. Let $\mathcal{C}_{\text{old\_avg}}$ denote the old score and $\mathcal{C}_{\text{new\_avg}}$ the updated ARF-scorer's output. The updated weight $\mathcal{C}_{\text{New}}$ is computed as:

$$\mathcal{S}_{\cos} = \sigma\left(\left(\text{CosSim}(\mathcal{C}_{\text{new\_avg}}, \mathcal{C}_{\text{old\_avg}}) - 0.5\right) \cdot S_{\text{sig}}\right) \quad (13)$$

$$\mathcal{C}_{\text{New}} = \mathcal{C}_{\text{new\_avg}} \cdot \mathcal{S}_{\cos} + \mathcal{C}_{\text{old\_avg}} \cdot (1 - \mathcal{S}_{\cos}) \quad (14)$$

This mechanism ensures that new scoring reflects both the current user preference and the historical decision boundary, enabling the ARF-scorer to retain knowledge of previous patterns rather than overfitting to recent feedback alone.

3.3 TraceBias Algorithm

With sufficient preference-enriched interaction data, we optimize policy behavior through a direct score-based actor-critic algorithm named **TraceBias**. While theoretically aligned with PPO and DPO (see Appendix B.3), TraceBias adopts a token-wise strategy and novel normalization mechanisms to enable stable, fine-grained reward-guided optimization—without relying on binary comparison labels.

The algorithm consists of two key components: (1) a **Double Average Method (DAM)** that normalizes satisfaction scores and token-level policy ratios, stabilizing training across variable-length sequences; and (2) an advantage function derived from trajectory-level score differences between fine-tuned and reference models, serving as the core optimization signal.

3.3.1 Double Average Method (DAM)

As part of the stabilization process, we correct gradient inconsistencies caused by varying sequence lengths. Rather than using explicit gradient clipping—which may discard useful signals—we apply two normalization strategies: one for satisfaction scores and one for token-level policy ratios.

Let T denote the sequence length, i index the sample, $x^{(i)}$ be the input context, and $y_t^{(i)}$ the t-th output token. P_θ and P_{old} represent the current and reference sample policy probabilities. We normalize token-level log-probabilities as follows, where $\pi_\theta^{(i)}$ and $\pi_{\text{old}}^{(i)}$ denote the average log-probabilities under the current and reference policies, and $\pi_{\text{ratio}}^{(i)}$ serves as the update ratio used in optimization:

$$\pi_\theta^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \log P_\theta(y_t^{(i)} \mid x^{(i)}) \quad (15)$$

$$\pi_{\text{old}}^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \log P_{\text{old}}(y_t^{(i)} \mid x^{(i)}) \quad (16)$$

$$\pi_{\text{ratio}}^{(i)} = \exp(\pi_\theta^{(i)} - \pi_{\text{old}}^{(i)}) \quad (17)$$

Through regularization of continuous scores via $\mathcal{C}_{\text{avg}}$ (see Eq. 8) and normalization of token-level policy ratios, these mechanisms jointly mitigate gradient imbalance—where longer sequences amplify log-prob norms and shorter ones weaken updates—and enhance gradient consistency more effectively than direct clipping, as confirmed by gradient norm analysis (Fig. 2).

3.3.2 Trace Scores with Discounted Step-wise Evaluation

After applying regularization to stabilize token-level optimization, we next compute the score difference between generated and reference trajectories—denoted as $\mathcal{S}_{\text{t-real}}$ and $\mathcal{S}_{\text{t-ref}}$, respectively—to estimate the advantage function.

At each step j, we compute a relative preference score based on the sentiment distribution:

$$\mathcal{S}^{(j)} = \mathcal{C}_{\text{avg[:2]}}^{(j)} - \mathcal{C}_{\text{avg[0]}}^{(j)} \quad (18)$$

where $\mathcal{C}_{\text{avg[:2]}}^{(j)}$ captures positive dimensions and $\mathcal{C}_{\text{avg[0]}}^{(j)}$ negative ones.

To aggregate these step-wise signals across a multi-turn trajectory, we apply a discount factor γ to compute trajectory-level scores, where $\mathcal{S}_{\text{ref}}^{(j)}$ and $\mathcal{S}_{\text{real}}^{(j)}$ denote the per-turn scores of the reference and generated responses at step j, respectively:

$$\mathcal{S}_{\text{t-real}} = \sum_{j=1}^{T} \gamma^{j-1} \cdot \mathcal{S}_{\text{real}}^{(j)}, \quad \mathcal{S}_{\text{t-ref}} = \sum_{j=1}^{T} \gamma^{j-1} \cdot \mathcal{S}_{\text{ref}}^{(j)} \tag{19}$$

The sentiment gap $\mathcal{S}_{\text{t-ref}}^{(i)} - \mathcal{S}_{\text{t-real}}^{(i)}$ then serves as an advantage signal guiding policy updates. This mechanism enables TraceBias to model evolving user sentiment over time, weighting earlier and later parts of a dialogue differently rather than treating outputs uniformly.

3.3.3 Final Representation of TraceBias

TraceBias integrates trajectory-level preference signals with token-level policy dynamics to construct its optimization objective. By combining trajectory-level advantage signals with DAM-normalized token-level policy ratios, TraceBias achieves stable and fine-grained reward-guided updates—without relying on binary preference labels. The final optimization objective $\mathcal{L}_{\text{final}}$ is expressed as:

$$\mathcal{L}_{\text{final}} = -\sum_{i=1}^{N} \left(\mathcal{S}_{\text{t-ref}}^{(i)} - \mathcal{S}_{\text{t-real}}^{(i)} \right) \cdot \pi_{\text{ratio}}^{(i)} \tag{20}$$

This objective underpins TraceBias's stable, fine-grained updates and provides a robust optimization foundation for the broader ARF framework.

4 Experiments

4.1 Experimental Setup

We evaluate ARF-RLHF framework and baseline RLHF methods on four lightweight LLMs: **Gemma2-2B**, **Qwen2-1.5B**, **Qwen2.5-1.5B**, and **LLaMA3.2-2B**. Each model is fine-tuned using one of the following RLHF approaches: our proposed **TraceBias**, which leverages trajectory-level bias and token-wise normalization without binary comparisons; **DPO**, which uses pairwise preference modeling via the Bradley-Terry framework; **PPO**, a scalar-reward-based policy optimization method; and **RLAIF**, which constructs comparison datasets automatically using LLMs.

To assess generalization across diverse tasks, we use five QA datasets from Big-Bench (Srivastava et al., 2023): **Alpaca** (Pawlik and Grigoriadis, 2024) (instruction following), **GSM8K** (Cobbe et al., 2021) (mathematical reasoning), **StrategyQA** (Geva et al., 2021) (commonsense QA), **TopicalChat** (Gopalakrishnan et al., 2023) (open-domain dialogue), and **CNN/DailyMail** (See et al., 2017) (summarization)—used primarily for robustness testing due to noise. Preference-labeled data is constructed via synonym substitution and scoring with pretrained models (see Appendix E).

We also introduce a multi-domain satisfaction dataset, **Emotion3**, comprising 78,630 samples aggregated from DailyDialog, GoEmotions, ISEAR (Scherer and Wallbott, 1997), and Sentiment140 (Go et al., 2009). Emotion labels are mapped to satisfaction levels and partially rescored using LLaMA3-13B and Qwen2.5-7B. Manual verification ensures label quality (details in Appendix H).

All models are fine-tuned using LoRA (Hu et al., 2022). Hyperparameter configurations are provided in Appendix A.

4.2 Main Results

Following the setup, we conduct targeted experiments to validate our framework through component-wise analysis, isolating the impact of key mechanisms via ablations and diagnostics. Below, we provide an overview of these core experiments.

- **Static Scorer:** Experiment 4.3 evaluates the reliability of the static satisfaction scorer used for preference collection, while Discussion 5.2 further substantiates its robustness.
- **ARF Scorer:** Experiment 4.4 examines how effectively the ARF scorer adapts to evolving preferences over time.
- **TraceBias Effectiveness:** Experiments 4.5 and 4.6 demonstrate that TraceBias yields more informative feedback signals under both human and LLM-generated preferences. We also added CaseStudy in appendix K.
- **Mechanism Validation:** Experiments 4.7, 4.8, and Experiments 4.9 highlight the role of key design components such as Experience Replay (ER), Dynamic Advantage Matching (DAM), and Preference-Biased Scoring.

4.3 Evaluation of the Static Satisfaction Scorer

The overall performance of the ARF framework strongly depends on the quality of the static satisfaction model used for initial data collection and supervision. While we describe the construction of

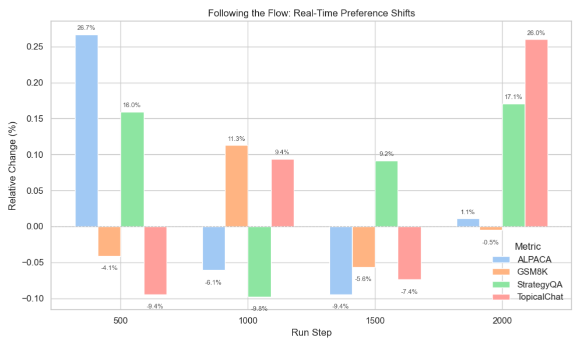

Figure 3: Tracking preference shifts using ARF. A drop in score indicates alignment with newly introduced negative preferences.

this model and its large-scale annotation base in Appendix H, its effectiveness must still be validated. We evaluate the scorer on five benchmark emotion classification datasets: DailyDialog, GoEmotions, ISEAR, Sentiment140, and Emotion3 (a merged set). As shown in Table 1, the model consistently achieves over 70% accuracy across all datasets.

4.4 Adaptive Preference Tracking via ARF

Leveraging the static scorer's accuracy, we test the ARF scorer's dynamic tracking by injecting bias-specific data every 500 steps in the order ALPACA→GSM8K→StrategyQA→TopicalChat. When more than two biases overlap, we apply negative supervision to the earliest bias. The resulting scoring shifts are shown in Figure 3.

The results demonstrate that ARF adapts effectively to changing preferences. Initially, we observe synchronized gain/loss patterns between ALPACA and StrategyQA, and between GSM8K and TopicalChat, likely due to semantic similarity. Despite this, ARF successfully distinguishes between tasks once negative preferences are introduced (e.g., ALPACA at step 1500), indicating its robustness to subtle semantic correlations.

4.5 RLHF Method Comparison under Unified Evaluation

Due to the instability and prompt sensitivity of AI-judge evaluations, we instead employ a unified reward model for both data filtering and evaluation. This removes variation from prompt design, sampling temperature, and model architecture(More in discussion 5.1). We compare PPO, DPO, and our TraceBias method under identical scoring supervision, across four tasks and four base models: Qwen2 1.5B, Qwen2.5 1.5B, LLaMA3.2 3B, and Gemma2 2B. The normalized performance relative

Dataset	DailyDialog	GoEmotions	ISEAR	Sentiment140	Emotion3
Accuracy (%)	70.05	73.65	76.00	74.10	71.60

Table 1: Test accuracy of the static satisfaction scorer on various sentiment datasets. Hyperparameter details are provided in Appendix 8.

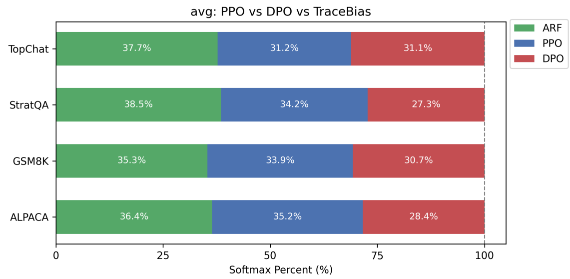

Figure 4: Average performance comparison of RLHF methods (PPO, DPO, TraceBias) under consistent scoring and preference targets. Single models' preformance in appendix J.

ratio compare to SFT is shown in Figure 4.

On average, TraceBias outperforms both PPO and DPO, with an improvement of 3.3% over PPO and 7.6% over DPO. We attribute this to the stability introduced by the DAM mechanism and the expressiveness of the trace-based update design. While there are isolated tasks where PPO or DPO perform better, TraceBias consistently ranks highest in aggregate performance.

4.6 LLM-Based Preference: RLAIF vs. ARF

To test TraceBias under machine-generated preferences, we construct a 1K preference dataset on StrategyQA using DeepSeek-v3(The detailed AI evaluation output in appendix F). Using this dataset, we train RLHF pipelines using PPO and DPO, denoted RLAIF-PPO and RLAIF-DPO. We compare them against TraceBias using the same reward supervision. As shown in Table 2, TraceBias outperforms both RLAIF variants, indicating its robustness to the quality of preference data.

Interestingly, while DPO slightly outperforms PPO in this setting, its dependence on precise comparisons makes it more sensitive to data quality. In contrast, TraceBias and PPO are better suited for

Evaluation method	RLAIF-PPO	RLAIF-DPO	TraceBias
Score Ratio	30.3	32.8	36.9
DeepSeek V3(win rate vs SFT)	43%	49%	52%

Table 2: The evaluation of multiple RLAIF variants against the TraceBias method on the StrategyQA dataset, using DeepSeek-v3 preference annotations.

ER Ratio	GSM8K (Preference)	Emotion3
Basic	53.52	73.84
0	60.59	59.32
0.5	56.40	70.88

Table 3: Ablation of ER ratio in ARF training. ER=0.5 balances adaptation and generalization.

Condition	**ALPACA**	**GSM8K**
With ReScore	-9.4%	-0.5%
Without ReScore	7.2%	3.7%

Table 4: Impact of rescoring on ALPACA and GSM8K after preference reversal. Negative values indicate successful adaptation.

noisy or weakly-supervised preference signals.

4.7 Effect of Experience Replay (ER) in ARF

We hypothesize that ER helps mitigate catastrophic forgetting or overfitting in ARF training. To test this, we compare ARF with and without ER under 1000 training steps. As shown in Table 3, disabling ER leads to better accuracy on recent data but a significant drop in generalization, supporting our claim.

4.8 Gradient Stability Analysis-DAM vs. Clip

As shown in Figure 2, we analyze gradient norms across PPO and TraceBias (using traceStep = 1 for fairness). TraceBias exhibits lower gradient magnitude and variance, even compared to PPO with clipping, supporting our claim that DAM promotes smoother and more stable learning dynamics.

4.9 On the Necessity of Rescoring in Preference-Biased Scoring

We conduct an ablation to evaluate the necessity of periodic rescoring during Adaptive Preference Tracking(Experiment 4.4). As shown in Table 4, disabling rescoring leads to increased scores even after preference reversal—indicating stale supervision and misalignment. In contrast, enabling rescoring correctly reflects negative feedback, reinforcing dynamic preference adaptation. This confirms rescoring as a critical mechanism for maintaining signal integrity in evolving reward landscapes.

5 Discussion

5.1 How to proof our experiments' accuracy?

As noted in Section 4.5, AI-agent–based evaluation (e.g., using an LLM judge) exhibits high variance from prompt wording, task quirks, model architecture, and random seeds, yielding inconsistent and unreliable results. To mitigate this, we complement AI-agent metrics (Section 4.6) and a unified, scorer-based protocol: for reward-oriented methods (e.g., TraceBias, PPO), we train against a pretrained reward model and evaluate with a shared, immutable scorer; for comparison-based methods (e.g., DPO), we use the same scorer to assess preference alignment. Crucially, every method and run uses the exact same held-out test examples (none seen during training), and the scorer never changes—eliminating any method-specific coupling or information leakage. This ensures a stable, unbiased comparison of each method's convergence to the target preference.

5.2 On the Reliability of Satisfaction Supervision

Satisfaction annotations are inherently subjective-labeling tasks involving large numbers of human annotators often reflect diverse preferences, even SOTA LLMs show bias when re-labeling Emotion3, with neutral predictions ranging from 24.0% to 37.3% (variance 29.51, Appendix G). Neutrality proves especially ambiguous, leading to unstable supervision. Although our static scorer reaches only 70% accuracy, it mitigates such uncertainty by excluding neutral scores during TraceBias updates (Eq. 18), using them instead to dampen noisy gradients e.g., Good: 0.02, Neutral: 0.90, Bad: 0.08 yields $S = -0.06$ resulting in minimal updates. This design—by ensuring softer updates for items dominated by neutral evaluations—avoids rigid binary comparisons under ambiguity, thereby improving robustness, reducing annotation variance, and enabling reliable tracking of individual preferences beyond crowd-level bias.

6 Conclusion

We introduce ARF-RLHF, a reinforcement learning framework aligning language models to user preferences. It includes an Adaptive Reward-Following scorer for satisfaction estimation, a perturbation-augmented preference generalization database, and TraceBias, an actor–critic optimization method with token-level stabilization. Theoretical analysis confirms compatibility with PPO/DPO, and experiments validate preference optimization under limited supervision.

References

RALPH ALLAN BRADLEY. 1955. Rank analysis of incomplete block designs: Iii. some large-sample results on estimation and power for a method of paired comparisons*. *Biometrika*, 42(3-4):450–470.

Huanpei Chen and Hsin-Hsi Chen. 2016. Implicit polarity and implicit aspect recognition in opinion mining. In *Annual Meeting of the Association for Computational Linguistics*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. Goemotions: A dataset of fine-grained emotions. *Preprint*, arXiv:2005.00547.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Preprint*, arXiv:2101.02235.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical Report CS224N Project Report, Stanford University.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-chat: Towards knowledge-grounded open-domain conversations. *Preprint*, arXiv:2308.11995.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Alastair Henry, Cecilia Thorsen, and Peter D. MacIntyre. 2021. Willingness to communicate in a multilingual context: part two, person-context dynamics. *Journal of Multilingual and Multicultural Development*, 42(9):827–841.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024b. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback. *Preprint*, arXiv:2309.00267.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Piotr Pawlik and Kristiana Grigoriadis. 2024. Alpaca-paper. Zenodo dataset.

Shrimai Prabhumoye, Sushant Choudhary, Eleni Spiliopoulou, Chris Bogart, Carolyn Rose, and Alan W Black. 2017. Linguistic markers of influence in informal interactions. In *Proceedings of the Second Workshop on NLP and Computational Social Science*, pages 53–62, Vancouver, Canada. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Klaus R. Scherer and Harald G. Wallbott. 1997. ISEAR dataset: International survey on emotion antecedents and reactions. https://www.unige.ch/cisa/research/materials-and-online-research/research-material/.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. 2017a. Trust region policy optimization. *Preprint*, arXiv:1502.05477.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017b. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

James Shanahan, Yan Qu, and Janyce Wiebe. 2006. *Computing Attitude and Affect in Text: Theory and Applications*, volume 20.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and 1 others. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

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

Rongzhi Zhang, Jiaming Shen, Tianqi Liu, Haorui Wang, Zhen Qin, Feng Han, Jialu Liu, Simon Baumgartner, Michael Bendersky, and Chao Zhang. 2024. PLaD: Preference-based large language model distillation with pseudo-preference pairs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15623–15636, Bangkok, Thailand. Association for Computational Linguistics.

Limitations

While our method offers significant theoretical and empirical advantages, we acknowledge the following limitations:

- **Lack of Real Huamn evaluation**: The scorer-based evaluation protocol is designed for fair comparison of RLHF methods, focusing on minimizing noise factors. It does not yet capture generalization performance in real-world scenarios. Future work will include human blind testing or cross-validation with alternative scorers to further validate the results.
- **Model Scale Constraint**: Due to resource constraints, we conduct experiments on 1.5B–3B models. While our method is architecture-agnostic and theoretically scalable, its effectiveness on larger LLMs (e.g., 7B, 13B, 65B) remains to be explored in future work.

A Hyper Parameters

All experiments' Hyper parameters shows below:

1. Static Scorer Evaluation: table 8
2. Adaptive Preference Tracking: table 9
3. RLHF Comparison: table 10
4. RLAIF vs. ARF: table 11
5. Effect of Experience Replay (ER) in ARF 12
6. Gradient Stability Analysis for DAM 13

All experiments were conducted on 2 * NVIDIA GTX 2080 Ti GPUs with CUDA unified memory enabled, using multi-GPU parallelism. Training consumed approximately 390 GPU-hours per device.

B The Essential Homology of DPO, PPO, and TraceBias

The Actor-Critic (AC) algorithm can be formulated as:

$$\mathcal{L}^{\mathrm{AC}}(\theta) = -\sum_{t=1}^{N} \log \pi_\theta(a_t \mid s_t) \cdot A_t \tag{21}$$

In this section, we demonstrate the theoretical connections among PPO, DPO, and our proposed TraceBias. We argue that these methods share a common optimization structure, thereby validating the theoretical soundness of TraceBias.

B.1 PPO as a clip Constrained Actor-Critic Variant

PPO (Schulman et al., 2017b) can be defined as:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min\left(r_t(\theta)A_t,\ \text{clip}(r_t(\theta), \epsilon)A_t\right)\right] \tag{22}$$

where $r_t(\theta)$ is the importance sampling ratio between the current and previous policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{23}$$

and the clipping function is defined as:

$$\text{clip}(r_t(\theta), \epsilon) = \begin{cases} 1-\epsilon, & \text{if } r_t(\theta) < 1-\epsilon \\ r_t(\theta), & \text{if } 1-\epsilon \le r_t(\theta) \le 1+\epsilon \\ 1+\epsilon, & \text{if } r_t(\theta) > 1+\epsilon \end{cases} \tag{24}$$

By expanding the objective, we obtain:

$$\mathcal{L}^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min\left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t, \text{clip} \left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \epsilon \right) A_t \right) \right] \tag{25}$$

If we ignore the clipping operation—which serves as a regularization mechanism to limit the policy update magnitude—the PPO objective reduces to:

$$\mathcal{L}^{\text{PPO}}(\theta) \propto \sum_{t=1}^{N} r_t(\theta) A_t \tag{26}$$

Here, $r_t(\theta)$ reflects the policy ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, which encourages increasing the likelihood of actions with high advantage A_t. This shows that PPO essentially shares the same optimization direction as AC, differing only in the incorporation of a trust-region-inspired constraint to stabilize training.

B.2 DPO as a KL-Constrained Actor-Critic Variant

Direct Preference Optimization (DPO) (Rafailov et al., 2024) leverages the Bradley-Terry model to represent pairwise preferences as follows:

$$\mathbb{P}(y^+ \succ y^- \mid x) = \frac{\exp(r(y^+))}{\exp(r(y^+)) + \exp(r(y^-))} \tag{27}$$

Its associated loss is:

$$\mathcal{L}_R(\phi, D) = -\mathbb{E}_{(x, y_w, y_l)} \left[\log \frac{\exp(r(y_w))}{\exp(r(y_w)) + \exp(r(y_l))} \right] \tag{28}$$

The DPO objective derived from this model is:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)\, \pi_{\text{ref}}(y_l \mid x)}{\pi_\theta(y_l \mid x)\, \pi_{\text{ref}}(y_w \mid x)} \right) \right] \tag{29}$$

Introducing a normalization constant $Z(x)$, the implicit optimal policy $\pi^*(y \mid x)$ can be defined as:

$$\pi^*(y \mid x) = \frac{\pi_{\text{ref}}(y \mid x) \cdot \exp(\frac{1}{\beta} r(y))}{Z(x)} \tag{30}$$

where the partition function $Z(x)$ is:

$$Z(x) = \sum_{y'} \pi_{\text{ref}}(y' \mid x) \cdot \exp\left(\frac{1}{\beta} r(y')\right) \tag{31}$$

Taking the logarithm of both sides yields:

$$\log \pi^*(y \mid x) = \log \pi_{\text{ref}}(y \mid x) + \frac{1}{\beta} r(x, y) - \log Z(x) \tag{32}$$

We can then derive:

$$r(x, y) = \beta \log \frac{\pi^*(y \mid x)}{\pi_{\text{ref}}(y \mid x)} + \beta \log Z(x) \tag{33}$$

By applying $r(x, y)$ in the pairwise preference model $\mathbb{P}(y^+ \succ y^- \mid x)$ and utilizing the Plackett-Luce model (see Appendix A.3 of (Rafailov et al., 2024) for more details), the DPO objective can be equivalently rewritten as:

$$\max_{\pi_\theta} \left\{ \mathbb{E}_{x\sim D,\, y\sim\pi_\theta(\cdot|x)} \left[r_\phi(x, y) \right] - \beta\, \text{KL}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\big) \right\} \tag{34}$$

Here, the optimization objective is to maximize the expected reward regularized by a KL divergence term. Assuming $A(x, y) = r(x, y)$, and temporarily ignoring the KL regularization, this reduces to an actor-critic style objective:

$$\mathcal{L}^{\text{DPO}}(\theta) \propto \mathbb{E}_{y\sim\pi_\theta(\cdot|x)} \left[A(x, y) \right] \tag{35}$$

This implies that DPO and actor-critic share effectively the same optimization objective when the reward signal is defined as the advantage. In practice, this expectation can be approximated by:

$$\mathcal{L}^{\text{DPO}}(\theta) \approx \sum_{i=1}^{N} \log \pi_\theta(y_i \mid x_i) \cdot A(x_i, y_i) \quad (36)$$

Thus, DPO can be interpreted as a KL-regularized actor-critic method, where the reward signal is derived from preference feedback rather than scalar returns.

B.3 TraceBias as a DAM-Constrained Actor-Critic Variant

We have previously outlined the Actor-Critic (AC) interpretation of TraceBias in our methodology. Here, we provide a formal derivation from the expanded formulation to its AC-style representation.

$$\mathcal{L}_{\text{final}} = -\sum_{i=1}^{T} (\mathcal{S}_{\text{t-ref}} - \mathcal{S}_{\text{t-real}}) \cdot \exp(\pi_\theta^{(i)} - \pi_{old}^{(i)}) \quad (37)$$

Following our methodology, the token-level reward (or *score*) is defined via discounted(γ) stepwise preferences as follows:

$$\mathcal{S}_{\text{t-real}} = \sum_{j=1}^{T} \gamma^{j-1} \cdot \mathcal{S}_{\text{real}}^{(j)}, \quad \mathcal{S}_{\text{t-ref}} = \sum_{j=1}^{T} \gamma^{j-1} \cdot \mathcal{S}_{\text{ref}}^{(j)} \quad (38)$$

Accordingly, we define the advantage function by measuring the difference between the real and reference trajectories:

$$\mathcal{A}_i = \mathcal{S}_{\text{t-ref}}^{(i)} - \mathcal{S}_{\text{t-real}}^{(i)} \quad (39)$$

Substituting this into the objective, TraceBias can be rewritten in an actor-critic form:

$$\mathcal{L}_{\text{TraceBias}} = \sum_{i=1}^{T} \mathcal{A}_i \cdot \exp(\pi_\theta^{(i)} - \pi_{old}^{(i)}) \quad (40)$$

To improve optimization stability, we introduce the DAM smooth surrogate strategy, which pools token-level scores and normalizes the policy ratio across the trajectory. Analogous to the clipping term in PPO and the KL regularization in DPO, DAM serves as a regularization mechanism: We define the token-level ratio as:

$$\pi_\theta^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \log P_\theta(y_t^{(i)} \mid x^{(i)}) \quad (41)$$

$$\pi_{\text{old}}^{(i)} = \frac{1}{T} \sum_{t=1}^{T} \log P_{\text{old}}(y_t^{(i)} \mid x^{(i)}) \quad (42)$$

Then, the normalized trajectory-level policy ratio is computed as:

$$\pi_{\text{ratio}}^{(i)} = \exp(\pi_\theta^{(i)} - \pi_{old}^{(i)}) \quad (43)$$

Thus, the final form of TraceBias is:

$$\mathcal{L}_{\text{TraceBias}} = \sum_{i=1}^{T} \mathcal{A}_i \cdot \pi_{\text{ratio}}^{(i)} \quad (44)$$

This derivation shows that TraceBias can be directly interpreted as an Actor–Critic method without introducing additional approximations, highlighting its theoretically grounded and streamlined formulation.

B.4 Summary

Although PPO, DPO, and TraceBias differ significantly in their final objectives, all can be reformulated as variants of the Actor-Critic (AC) framework. By optimizing the expected reward weighted by advantage, each method introduces distinct regularization strategies—such as PPO's clipping, DPO's KL constraint, or TraceBias's DAM normalization—to improve training stability. This unified perspective highlights that TraceBias is not only theoretically grounded but also competitive with existing policy optimization techniques.

C The pseudo code of TraceBias

The TraceBias pseudo code shows in alg 1.

D The selection of sigmoid Scale

Scale plays a crucial role in determining the sensitivity of the norm function, as it directly affects most parameters. We recommend selecting scales within the range of [4,8], as excessively large values can push most parameters towards the boundaries of the Sigmoid function interval, limiting their effective range. Table 5 illustrates when $(\text{CosSim}(\mathcal{C}_{\text{ARF\_avg}}, \mathcal{C}_{\text{basic\_avg}}) - 0.5)$ equals values 0.2 is converted together with the corresponding function value range by the Sigmoid function under different proportional settings. It is worth noting that when the scale is set to 1, the relatively large 0.2 level in sigmoid results in a mapping value of only 0.5498. However, when the ratio increases to 16, the same input is mapped to nearly 1, indicating

Scale value	Scale Impact on Sigmoid under 0.2	interval
1(Too small)	0.5498	[0.3775,0.6225]
4	0.6900	[0.1192,0.8808]
6	0.7685	[0.0474,0.9526]
8	0.8320	[0.018,0.9820]
16(Too big)	0.9608	[0.0003,0.9997]

Table 5: The table shows how different scale values map to specific intervals after applying the Sigmoid function.

Evaluation method	Win	Loss	Equal
RLAIF-PPO	43%	50%	7%
RLAIF-DPO	49%	47%	4%
TraceBias	52%	44%	4%

Table 6: The win, loos, equal rate compare to SFT method

Model	Positive (%)	Neutral (%)	Negative (%)
Actual Samples (Uniform)	33.3	33.3	33.3
GPT-4o	40.3	30.3	29.3
DeepSeek-v3	44.0	24.0	32.0
Gemini 2.0 Flash	35.0	37.3	27.6
Variance	13.64	29.51	13.28

Table 7: Label distribution variance across models on Emotion3 samples. High neutral sentiment divergence highlights inconsistency in satisfaction estimation.

that the range is overly compressed. Based on these observations, we strongly recommend selecting a scale within the range of [4,8] for the numerical deviation annotation algorithm, as it ensures a balanced transformation without pushing values to extremes.

E RLHF Dataset Construction

To support comparison-based fine-tuning methods such as DPO and PPO, we construct a simulated binary preference dataset. Given the prohibitive cost of large-scale human annotation, and the fact that this dataset is primarily used to compare fine-tuning preferences across methods rather than for real-world deployment, we adopt a surrogate construction strategy that also aligns with the training of our ARF preference model.

Concretely, we employ the `naw.SynonymAug` module from the `nlpaug` library to perform four rounds of synonym substitution using WordNet, generating paraphrased variants that preserve semantic intent while introducing surface-level diversity.

Before constructing the binary comparison dataset, we annotate the augmented **'former'** samples with soft labels using our fine-tuned static satisfaction scorer. To amplify preference signals and avoid potential overfitting, we prepend task-specific prompts that were never included in the scorer's training data. Specifically:

- **Good prompt**: `Great! You gave a correct answer. Here is the next question: ...`
- **Bad prompt**: `Your answer is absolutely wrong! This is the next question. Stop giving such terrible and misleading feedback! ...`

The annotated samples are then used to fine-tune the ARF scorer, which serves as a proxy for user-aligned preferences. To ensure fairness, each RLHF method (including PPO and DPO) constructs its training pairs using the same ARF scorer: for each pair, the sample with a higher score is designated as the preferred (positive) response, while the lower-scored one is treated as negative. This guarantees that all methods are aligned in their optimization direction and evaluated under consistent supervision.

F DeepSeek agent's Evaluation under RLAIF task

We using below prompt to compare the output of baselines and SFT, the win/loos/evqual tabel shows in tabel 6. The prompt of comparison shows below:

`Question: '...' Answer1: '...' Answer2: '...' Please use strict criteria to determine which answer is more in line with human preferences 1 or 2 only answer a number.`

G Table of Subjective Experiments

The table 7 reveals that even state-of-the-art LLMs exhibit significant variance in emotion judgment, especially in distinguishing neutral sentiment—underscoring the inherent noise in satisfaction estimation.

H Satisfaction Dataset Construction

To construct a large-scale, diverse, and high-quality satisfaction classification dataset aligned with our three-level labeling schema (*bad*, *neutral*, *good*), we aggregate a total of 78,630 samples from four widely-used emotion and sentiment datasets:

- **DailyDialog** (Li et al., 2017): A multi-turn dialogue dataset that closely mirrors everyday

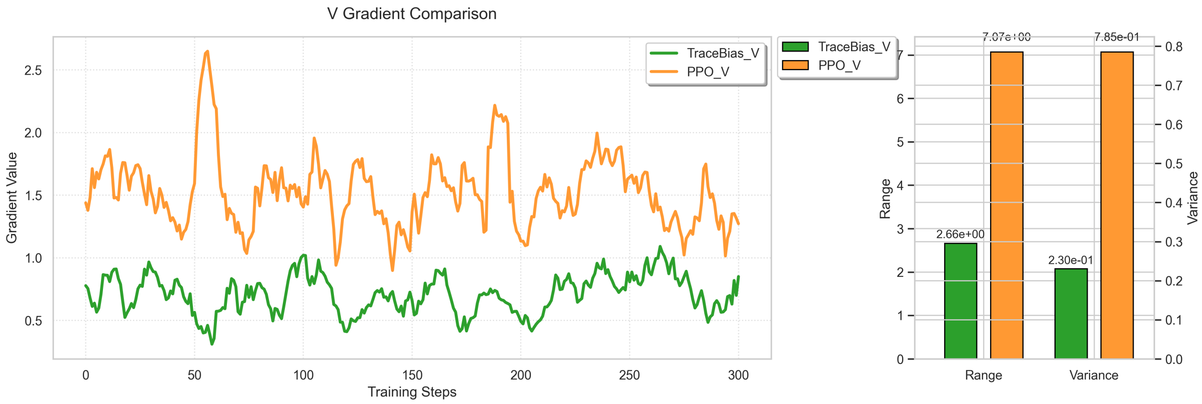

Figure 5: V Gradient norm comparison between PPO (with clip range $\epsilon = 0.2$) and TraceBias with DAM.

conversational scenarios.

- **GoEmotions** (Demszky et al., 2020): A fine-grained, high-quality emotion classification dataset spanning a wide range of affective states.
- **ISEAR** (Scherer and Wallbott, 1997): A clean and structured emotion dataset based on psychological self-reports.
- **Sentiment140** (Go et al., 2009): A large-scale Twitter sentiment dataset that reflects informal and noisy online communication.

To unify the labeling across datasets with heterogeneous annotation schemes, we define a common strength-based mapping strategy, converting existing emotion tags into a standardized 7-level satisfaction scale (see Table 14). For relatively clean datasets (DailyDialog, GoEmotions, ISEAR), we directly apply this mapping to assign satisfaction scores.

Given the informal nature of Sentiment140, additional cleaning is necessary. We sample 15,000 instances and perform multi-round evaluation using both Qwen2 7B and LLaMA3 13B. Each sample is scored twice by each model; the maximum and minimum scores are discarded, and the mean of the remaining two is taken as the final label. Samples with high variance across scores are further manually verified to ensure annotation reliability. The result is a cleaned subset of 15,000 samples from Sentiment140 with stable satisfaction labels.

After consolidating all datasets, we create a unified **Emotion7** dataset with 7 satisfaction levels. We then perform a coarse mapping to form the final **Emotion3** dataset: levels $[0, 1]$ as *bad*, 1.5 as *neutral*, and $[2, 3]$ as *good*. This dataset provides broad domain coverage, consistent labels, and stylistic diversity, serving as the basis for training our static satisfaction scorer.

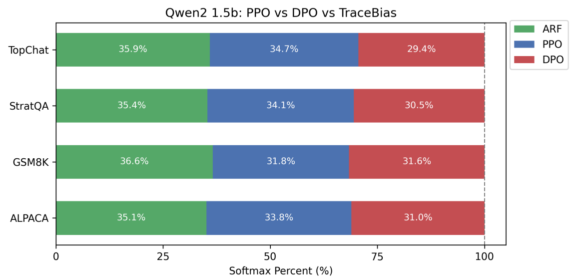

Figure 6: Qwen2's Performance

I Gradient Comparison V

Figure 5 V Gradient norm comparison between PPO (clip) and TraceBias (DAM). Lower variance and norm suggest improved stability.

J Models' Performance under different RLHF Baselines

We shows all models' RLHF performance below:

- Qwen2-1.5B: table 6
- Qwen2.5-1.5B: table 7
- Gemma2-2B: table 8
- Llama3.2-3B: table 9

We applied softmax with temperature (set to 0.1) purely for visualization purposes.

K CaseStudy under Llama3.2

To preserve the original formatting of model outputs, we retain their format in the paper. For excessively long responses, we replace parts with ellipses ("...") for clarity. Representative examples are provided in Table 15, Table 16, Table 17, and Table 18.

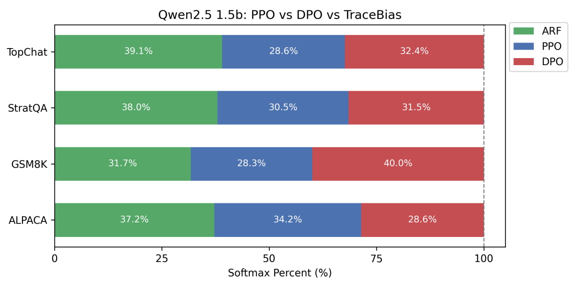

Figure 7: Qwen2.5's Performance

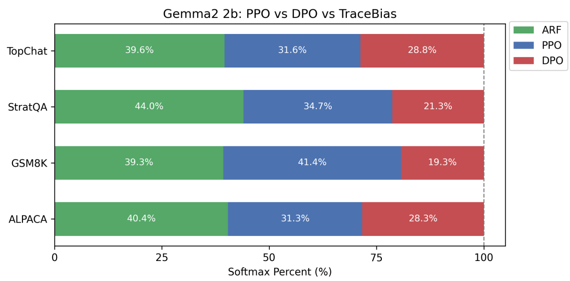

Figure 8: Gemma2's Performance

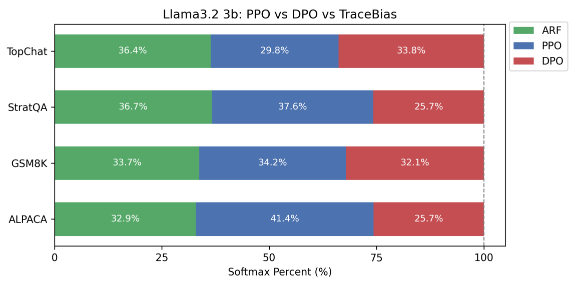

Figure 9: Llama3.2's Performance

Algorithm 1: TraceBias with DAM

Input: Augmented dialogue dataset $\mathcal{D}_{\text{aug}}$, reward model $\mathcal{R}$, policy model π_θ, reference model π_{ref}, max length $L_{\max}$, discount factor γ, environment $\mathcal{E}$

Output: Updated parameters θ

while *training* **do**
  repeat
    Sample a dialogue trajectory $\{(x_t, y_t^*, s_t)\}_{t=1}^{T} \sim \mathcal{D}_{\text{aug}}$;
    Initialize context $\mathcal{C} \leftarrow$ `[system prompt]`, total loss $\mathcal{L} \leftarrow 0$;
    Initialize `LabelReward` $\leftarrow 0$, `RealReward` $\leftarrow 0$;
    for $t = 1$ **to** T **do**
      Append user input x_t to context $\mathcal{C}$;
      `LabelReward` $+= \gamma^{t-1} \cdot s_t$;
      Generate output $y_t \sim \pi_\theta(\cdot \mid \mathcal{C})$;
      if *length*$(\mathcal{C}) > L_{\max}$ **then**
        break
      `// Compute per-token log-probabilities from both models`
      $\log \pi_\theta =$ `log_softmax`$(\pi_\theta(\mathcal{C}))$;
      $\log \pi_{\text{ref}} =$ `log_softmax`$(\pi_{\text{ref}}(\mathcal{C}))$;
      `// Compute average token-level log-probs (DAM token -level average)`
      $\log \pi_\theta(y_t) = \frac{1}{|len(y_t)|} \sum_{i=1}^{|y_t|} \log \pi_\theta(y_t^{(i)} \mid \mathcal{C})$;
      $\log \pi_{\text{ref}}(y_t) = \frac{1}{|len(y_t)|} \sum_{i=1}^{|y_t|} \log \pi_{\text{ref}}(y_t^{(i)} \mid \mathcal{C})$;
      `// Compute real reward via environment`
      $r_t \leftarrow \mathcal{E}(y_t)$;
      `RealReward` $+= \gamma^{t-1} \cdot r_t$;
      `// Importance sampling ratio`
      $w_t = \exp\left(\log \pi_\theta(y_t) - \log \pi_{\text{ref}}(y_t)\right)$;
      `// Add weighted reward mismatch to loss`
      $\mathcal{L} += -w_t \cdot ($`LabelReward` $-$ `RealReward`$)$;
      Append assistant response y_t to context $\mathcal{C}$;
  until valid sample obtained;
  `// Gradient update`
  Backpropagate: $\nabla_\theta \mathcal{L}$;
  Update: $\theta \leftarrow \theta - \eta \cdot \nabla_\theta \mathcal{L}$;

Hyper Parameters	TrainStep	Test Step	Learning Rate	Batch	MLP Hidden Size
Static Scorer	20000	500	1e-6	20	328

Table 8: hyper-parameters of Static Scorer Evaluation

Hyper Parameters	TrainStep	Test Step	Learning Rate	ERRatio	MLP Hidden Size
ARF Scorer	2000	500	1e-6	0.5	328

Table 9: hyper-parameters of Adaptive Preference Tracking via ARF

Hyper Parameters	LoRA Rank	Epoch	Test Step/Epoch	Train Step/Epoch	Learning Rate	TraceBias gamma	clip epsilon	PPO [c1,c2]	DPO beta
PPO	8	4	100	500	1e-6	-	0.2	[0.01,0.01]	-
DPO	8	4	100	500	1e-6	-	-	-	0.1
TraceBias	8	4	100	500	1e-6	0.99	-	-	-

Table 10: hyper-parameters of Adaptive Preference Tracking via ARF

Hyper Parameters	LoRA Rank	Epoch	Test Step/Epoch	Train Step/Epoch	Learning Rate	TraceBias gamma	clip epsilon	PPO [c1,c2]	DPO beta
PPO	8	2	100	250	1e-5	-	0.2	[0.01,0.01]	-
DPO	8	2	100	250	1e-5	-	-	-	0.1
TraceBias	8	2	100	250	1e-5	0.99	-	-	-

Table 11: hyper-parameters of RLAIF vs. ARF

ERRatio	TrainStep	Test Step	Learning Rate	MLP Hidden Size
0.5	1000	500	1e-5	328
0	1000	500	1e-5	328

Table 12: hyper-parameters of Experience Replay (ER) in ARF

Hyper Parameters	LoRA Rank	Train Step	Learning Rate	TraceBias gamma	clip epsilon	PPO [c1,c2]
PPO	8	300	1e-6	-	0.2	[0.01,0.01]
TraceBias	8	300	1e-6	0.99	-	-

Table 13: hyper-parameters of Gradient DAM test

Strength	Label	DailyDialog	ISEAR	GoEmotion
0.0	extremely neg	disgust	disgust	disappointment embarrassment grief sadness
0.5	neg	anger, sadness	sadness	anger disgust remorse
1.0	neg nero	fear	fear	annoyance confusion disapproval nervousness
1.5	nero	no emotion		caring curiosity neutral
2.0	pos nero			approval desire realization relief surprise
2.5	pos	surprise	joy	admiration amusement excitement joy love optimism pride
3.0	extreme pos	happiness		admiration gratitude

Table 14: Emotion categories mapped to continuous strength scores across datasets.

Table 15: GSM8K

Field	Content
Prompt	Q: Reggie and his brother are having a basketball shooting contest. They each get to take 10 shots. Layups are worth 1 point, free throws are worth 2 points, and anything further away is worth 3 points. Reggie makes 3 layups, two free throws, and one long shot. His brother only shoots long shots and makes 4 of them. How many points does Reggie lose by? A:
label	Reggie makes 3 points off layups because 3 × 1 = <<3*1=3>>3 Reggie makes 4 points off free-throws because 2 × 2 = <<2*2=4>>4 Reggie makes 3 points off long shots because 1 × 3 = <<1*3=3>>3 Reggie scores 10 points because 3 + 4 + 3 = <<3+4+3=10>>10 His brother scores 12 points because 4 × 3 = <<4*3=12>>12 Reggie loses by 2 because 12 minus 10 = <<12-10=2>>2 2
SFT	To find out how many points Reggie lost by, we calculate both totals explicitly: Reggie's points: layups 3 × 1 = 3; free throws 2 × 2 = 4; long shot 1 × 3 = 3; sum = 10. Brother's points: 4 × 3 = 12. Since 10 < 12, Reggie loses by 2 points.
TraceBias	To determine the points Reggie scored, we need to calculate the points from each type of shot. Reggie made 3 layups (3 × 1 = 3 points), 2 free throws (2 × 2 = 4 points), and 1 long shot (1 × 3 = 3 points). Total: 3 + 4 + 3 = 10 points. His brother made 4 long shots (4 × 3 = 12 points). Difference: 12 minus 10 = 2 points.
Comment	The TraceBias response employ formal mathematical notation and rigorous step-by-step reasoning to enhance precision and clarity.

Table 16: ALPACA

Field	Content
Prompt	Q: Describe the character of Romeo from Romeo and Juliet. A:
label	"Romeo is a passionate, impulsive, and hopelessly romantic young man, who falls deeply in love with Juliet at first sight. He is a member of House Montague in the play *Romeo and Juliet*, and is depicted as sensitive, idealistic, and poetic, idolizing love and its power. Although he's brave and skilled in combat, Romeo is also shown as prone to intense emotions, leading him to act impulsively and recklessly. His strong feelings often cloud his judgement, which eventually leads to the tragic end of the play. Despite his flaws, Romeo is a loyal friend, devoted son and nephew, and an unwavering lover, willing to do anything for the woman he loves."
SFT	Romeo Montague is the protagonist of William Shakespeare's tragic play *Romeo and Juliet*. He is a young, passionate, and impulsive nobleman from the rival Montague family. 1. **Romantic and passionate**: falls in love at first sight and sacrifices everything for Juliet. 2. **Impulsive and impetuous**: acts on emotion without forethought. 3. **Loyal and devoted**: fiercely protects family and Juliet. 4. **Youthful and inexperienced**: mistakes due to naiveté. 5. **Emotionally expressive**: conveys feelings through poetry. 6. **Self-destructive**: reckless decisions contribute to tragedy. His tragic fate underscores the destructive power of hate and the redemptive power of love.
TraceBias	Romeo, the protagonist of Shakespeare's tragic love story, is a complex, passionate, and impulsive nobleman from the House of Montague. **Key Characteristics:** (1) *Impulsive and passionate*; (2) *Romantic and idealistic*; (3) *Impetuous*; (4) *Loyal and devoted*. **Motivations:** love for Juliet, family loyalty, desire for vengeance. **Flaws:** impulsivity, lack of prudence, emotional instability. His character symbolizes love's power and the human spirit.
Comment	The TracceBias response is thorough with the format more close to label.

Table 17: StrategyQA

Field	**Content**
Prompt	Q: Did compact discs make computer gaming more popular? A:
label	Yes. Compact discs contained significantly more storage space than the previously popular floppy disc format. Gaming studios were therefore able to significantly improve the graphics, sounds, and features of their games to make them more immersive. The better games led to a massive increase in popularity for computer gaming.
SFT	Compact discs (CDs) did play a role in making computer gaming more popular, but it was not the sole factor. • **CD-ROM technology**: enabled large-game distribution on a single disc. • **Improved graphics and sound**: enhanced audio-visual experience (e.g. *Doom*, *Myst*). • **Wider availability**: expanded platform reach. • **Industry growth**: attracted investment and talent. Other influences included hardware advances, online gaming, and marketing strategies.
TraceBias	Yes. CDs significantly popularized PC gaming in the 1990s. The CD-ROM format's increased capacity and audio quality drove a surge in game sales, fueling industry growth.
Comment	The SFT response is verbose and includes peripheral historical details; TraceBias more concise focus on the primary mechanisms by which CD technology impacted gaming would enhance clarity and relevance.

Table 18: TopicalChat

Field	**Content**
Prompt	agent_B: "Did you know Steve Spurrier coach the Redskins?" … agent_B:
label	"he has 12 victories in 32 games so maybe thats why he got fired , I think the team had high hopes on him."
SFT	It appears that agent_A and agent_B are discussing Steve Spurrier's career: 1. Confirmed that Spurrier coached the Redskins. 2. Recounted anecdote about his duck-lips TV moment. 3. Noted his 1966 Heisman Trophy at Florida. 4. Identified misattribution of Tampa Bay Bandits. 5. Speculated on team embarrassment and website omission.
TraceBias	agent_B: "I know, its hard to find characters as known and famous as Steve lol..." agent_A: "I think you're right. I mean, Steve Spurrier ... need to give him the recognition he deserves."
Comment	The SFT submission fails to recognize the continuation task and instead provides a summary, whereas the TraceBias submission demonstrates proper dialogue continuation and contextual sensitivity.