PERSA: Reinforcement Learning for Professor-Style Personalized Feedback with LLMs

Ravi Ranjan*
Florida International University (FIU), Miami, USA
rkuma031@fiu.edu

Utkarsh Grover
University of South Florida (USF), Tampa, USA
utkarshgrover@usf.edu

Xiaomin Lin
University of South Florida (USF), Tampa, USA
xlin2@usf.edu

Agoritsa Polyzou*
Florida International University (FIU), Miami, USA
apolyzou@fiu.edu

Abstract

Large language models (LLMs) can provide automated feedback in educational settings, but aligning an LLM's **style** with a specific instructor's tone while maintaining diagnostic correctness remains challenging. We ask: *how can we update an LLM for automated feedback generation to align with a target instructor's style without sacrificing core knowledge?* We study how Reinforcement Learning from Human Feedback (RLHF) can adapt a transformer-based LLM to generate programming feedback that matches a professor's grading voice. We introduce **PERSA**, an RLHF pipeline that combines supervised fine-tuning on professor demonstrations, reward modeling from pairwise preferences, and Proximal Policy Optimization (PPO), while deliberately constraining learning to **style-bearing components**. Motivated by analyses of transformer internals, PERSA applies parameter-efficient fine-tuning. It updates only the *top* transformer blocks and their feed-forward projections, minimizing global parameter drift while increasing stylistic controllability. We evaluate our proposed approach on three code-feedback benchmarks (APPS, PyFiXV, and CodeReviewQA) using complementary metrics for style alignment and fidelity. Across both Llama-3 and Gemma-2 backbones, PERSA delivers the strongest professor-style transfer while retaining correctness; for example, on APPS, it boosts Style Alignment Score (SAC) to 96.2% (from 34.8% for Base) with Correctness Accuracy (CA) up to 100% on Llama-3 and Gemma-2. Overall, PERSA offers a practical route to personalized educational feedback by aligning both *what it says* (content correctness) and, crucially, *how it says it* (instructor-like tone and structure).

1 Introduction

Large language models (LLMs) are increasingly deployed in real-world systems, powering applications ranging from conversational assistants to code-generation tools. Major organizations such as OpenAI, Google, and Microsoft have emphasized alignment and prompt engineering to ensure these systems respond in useful and safe ways. For example, models such as ChatGPT and GPT-4 are refined through reinforcement learning from human feedback (RLHF) to better follow instructions and safety preferences (Ouyang et al., 2022; Chaudhari et al., 2025), while instruction-tuning pipelines are widely used to steer tone and response style in user-facing systems (Wang et al., 2023b; Bhattacharya et al., 2024). Although these efforts have substantially improved general helpfulness and harmlessness, far less attention has been paid to *personalized stylistic alignment*, where a model must communicate in a manner consistent with a particular human expert. This gap is especially important in education.

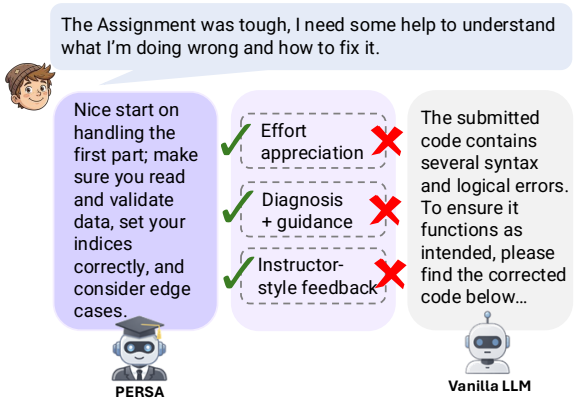

Figure 1: Illustration of how PERSA transforms generic LLM feedback into instructor-style, constructive, and actionable guidance compared to a vanilla LLM.

One promising use of LLMs in education is automated feedback generation. Prior work suggests that LLM-based tutors can provide hints and critiques in domains such as programming education (Razafinirina et al., 2024). However, the effectiveness of such feedback depends not only on correctness, but also on *how* it is delivered. Research in

*Corresponding authors.

Proceedings of the 21st Workshop on Innovative Use of NLP for Building Educational Applications, pages 529–545

the learning sciences shows that tone, phrasing, and instructional persona strongly affect how students interpret and respond to feedback (Demszky et al., 2024; Loughran, 2019). Instructors often develop a recognizable style that balances encouragement with precision, emphasizing issues such as readability, correctness, and edge-case reasoning (Felder et al., 1988). Feedback that is technically correct but overly generic or terse may therefore be perceived as less trustworthy, less supportive, and less pedagogically useful (Carless, 2012; Zhang and Hyland, 2022). Aligning AI-generated feedback with an instructor's authentic voice could make automated tutoring more acceptable and educationally effective.

In this work, we ask: *Can a pre-trained LLM be adapted to emulate a specific instructor's feedback style while preserving its problem-solving ability?* Existing instruction-tuned models are typically optimized for broad helpfulness and politeness, but alignment to an individual educator's style remains underexplored. To address this, we propose **Professor-Style Reinforcement-based Style Adaptation (PERSA)**, a method that uses RLHF to align model outputs with the feedback style of a target professor. Using a dataset of instructor-authored feedback on student work, we train a reward model that captures stylistic preferences and optimize the policy for *stylistic fidelity* rather than generic user satisfaction. As shown in Figure 1, whereas a vanilla LLM might generate a brief suggestion such as "Check your input handling," professor-style feedback is typically more explanatory, supportive, and actionable. PERSA narrows this gap by producing feedback that more closely reflects the instructor's voice, including encouragement, diagnosis, corrective guidance, and verification-oriented follow-up. In doing so, it supports seamless personalization of AI tutors for specific courses and teaching contexts.

A central technical contribution of PERSA is a *style-targeted, parameter-efficient fine-tuning strategy*. Rather than updating all model weights, which is computationally costly and risks degrading core capabilities, we focus adaptation on components most relevant to stylistic expression. Prior work suggests that many style-related properties are concentrated in higher transformer layers (Lai et al., 2024; Koto et al., 2022). Guided by this observation, we use low-rank adaptation (Hu et al., 2022) to fine-tune only the final layers of a 3B-parameter LLM. This selective update strategy preserves the model's underlying problem-solving competence while shifting its communicative "voice," reducing the catastrophic forgetting that can arise from indiscriminate fine-tuning (Luo et al., 2023). As a result, PERSA captures subtle instructor-specific traits, such as level of detail and encouraging tone, without compromising content accuracy.

We evaluate PERSA on a programming feedback generation task against strong baselines, including the base pre-trained model, supervised fine-tuning on the same instructor data, and an InstructGPT-style RLHF model optimized for general feedback quality. PERSA achieves the best overall balance between style and correctness. Its outputs show strong stylistic alignment with professor feedback, on Gemma-2 model with a Style Alignment Score approaching 0.99, and BLEU overlap above 98%, substantially exceeding competing methods (Papineni et al., 2002; Fu et al., 2018). At the same time, correctness remains at 100%, and feedback retains a consistently polite and constructive tone as measured by a politeness model (Danescu-Niculescu-Mizil et al., 2013). Human evaluators also strongly prefer PERSA, often describing its responses as reading "like they came from the professor." These findings suggest that combining RLHF with targeted layer adaptation is effective for capturing nuanced pedagogical style beyond what standard fine-tuning or generic alignment methods can achieve.

In summary, this paper makes three contributions: **(1)** we introduce **PERSA**, an RLHF-based framework for aligning LLM feedback with a specific instructor's pedagogical style; **(2)** we develop a *layer-selective* LoRA-based adaptation strategy that achieves stylistic transformation with minimal disruption to the model's original capabilities; and **(3)** we provide comprehensive evaluation, including automated style metrics and a human study, showing that PERSA produces feedback that is both pedagogically authentic and factually correct. Overall, this work advances the development of AI tutors that can not only provide accurate guidance but also communicate with the tone and effectiveness of a human instructor.

2 Related Work

Educational Feedback and Pedagogical Style. Prior work in educational technology shows that the *manner* of feedback delivery, including tone, politeness, and personalization, substantially affects learner engagement and outcomes (Han et al., 2023; Sonlu et al., 2024). Recent studies on LLMs

in education similarly emphasize that encouragement, specificity, and perceived instructor authenticity influence student trust and uptake (Razafinirina et al., 2024; Wu et al., 2024a). While this connects to broader alignment goals in dialogue systems (Alsafari et al., 2024; Wang et al., 2023b), our focus is narrower: *pedagogical alignment*, i.e., adapting model outputs to an instructor's communicative style rather than optimizing only for generic helpfulness (Bhattacharya et al., 2024; Matarazzo and Torlone, 2025).

Automated Feedback Generation. Automated programming feedback has long been studied through rule-based, analysis-driven, and repair-based methods. Keuning et al. provide a foundational survey of these approaches and their evaluation limitations (Keuning et al., 2018), while *Pedal* demonstrates practical code-aware feedback generation at scale (Gusukuma et al., 2020). More recently, LLMs have shown promise but also reliability concerns: GPT-4 can generate readable feedback yet still requires validation (Dai et al., 2024); MOOC-scale studies report that LLM feedback helps detect errors but is often inaccurate without test-based grounding (Gabbay and Cohen, 2024); and performance drops further in specialized settings such as concurrency debugging (Estévez-Ayres et al., 2024). Other efforts explore diversified repair generation (Choi and Lee, 2025) and responsible deployment considerations such as inclusivity and hallucination risk (Lindsay et al., 2025). Collectively, these works highlight that existing systems still under-address instructor-consistent tone and specificity.

RLHF for Language Model Alignment. RLHF has become a standard paradigm for aligning LLMs to human preferences (Ouyang et al., 2022; Xie et al., 2024). InstructGPT established the now-common pipeline of supervised fine-tuning, reward modeling, and PPO-based policy optimization (Ouyang et al., 2022), and subsequent work has extended RLHF to capture subtler properties such as tone, framing, and politeness (Kamath et al., 2024; Kirk et al., 2023; Dong et al., 2024; Yan et al., 2024). This is particularly relevant in education, where instructor judgments define high-quality feedback. However, full-parameter RLHF is computationally expensive and can unintentionally alter broader model capabilities (Chaudhari et al., 2025). Parameter-efficient alternatives, including LoRA-augmented RLHF and related methods, reduce this cost while preserving alignment quality (Sidahmed et al., 2024; Hong et al., 2024; Ethayarajh et al., 2024; Hu et al., 2022; Dettmers et al., 2023; Wu et al., 2024b). Importantly, prior analyses suggest that RLHF-induced changes are concentrated in upper transformer layers, especially FFN components (Moradi et al., 2024), motivating our layer-selective adaptation strategy.

Style Adaptation in Language Models. Style control in LLMs has been explored through prompting, style tokens, and fine-tuning, with recent work moving toward neuron- and layer-level analysis. Lai et al. show that stylistic attributes are disproportionately localized in higher transformer layers and FFN submodules, enabling targeted control via style-specific neurons (Lai et al., 2024; Shi et al., 2024). Related studies further suggest that selectively tuning or freezing layers can outperform full-model adaptation for certain alignment objectives (Shi et al., 2024; Wang et al., 2023a). Parameter-efficient methods such as LoRA and QLoRA make such targeted adaptation practical, though quantized tuning may introduce accuracy trade-offs in some settings (Dettmers et al., 2023; Hu et al., 2022; Ozdemir, 2023). In contrast to prior style-control work, PERSA applies these insights to *educational feedback*, aligning LLM outputs to an individual instructor's pedagogical voice while preserving correctness.

3 Preliminaries

Transformer LLMs as policies. We consider a decoder-only transformer language model as a stochastic policy that generates a feedback sequence $y = (y_1, \ldots, y_T)$ token-by-token conditioned on a prompt x (e.g., problem statement + student solution).

Data: demonstrations and preferences. We assume (i) an instructor demonstration set $\mathcal{D}_{\text{SFT}} = \{(x_i, y_i^\star)\}_{i=1}^{N}$ with professor-written feedback $y_i^\star$, and (ii) a preference set $\mathcal{D}_{\text{pref}} = \{(x_j, y_j^{(w)}, y_j^{(l)})\}_{j=1}^{M}$, where $y^{(w)}$ is preferred to $y^{(l)}$ under instructor style (and typically correctness). Preference data is used to learn a reward model for RLHF (Ouyang et al., 2022).

Supervised fine-tuning (SFT). SFT initializes the policy by minimizing the teacher-forcing negative log-likelihood:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\,\mathbb{E}_{(x,y^\star)\sim\mathcal{D}_{\text{SFT}}}\left[\sum_{t=1}^{|y^\star|} \log \pi_\theta\left(y_t^\star \mid x, y_{<t}^\star\right)\right]. \tag{1}$$

Here $y^{\star}_{<t}$ is the the gold prefix tokens before step t.

Reward modeling from pairwise comparisons. RLHF learns a scalar reward function $r_\phi(x, y) \in \mathbb{R}$ such that preferred responses receive higher reward. Using the Bradley–Terry model, the probability that $y^{(w)}$ is preferred over $y^{(l)}$ is

$$\Pr(y^{(w)} \succ y^{(l)} \mid x) = \sigma\big(r_\phi(x, y^{(w)}) - r_\phi(x, y^{(l)})\big), \tag{2}$$

yielding the standard pairwise logistic loss, and σ is the sigmoid function (Ouyang et al., 2022).

Policy optimization with PPO and KL control. Given r_ϕ, policy optimization seeks high reward while constraining drift from a reference policy π_{ref} (often the post-SFT model) via a KL penalty.

We optimize this objective with Proximal Policy Optimization (PPO) (Schulman et al., 2017). Let $s_t = (x, y_{<t})$ be the state (prefix context), $a_t = y_t$ the action (next token), and $\rho_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$.

Parameter-efficient adaptation with LoRA. To preserve base model knowledge and reduce computation, PERSA adopts Low-Rank Adaptation (LoRA) (Hu et al., 2022). For a frozen weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA learns a low-rank update ΔW with $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$ and $r \ll \min(d, k)$:

$$W' = W + \Delta W, \quad \text{where } \Delta W = BA. \tag{3}$$

We denote the trainable adapter parameters by θ_{LoRA} and treat the remaining parameters as frozen. Parameter-efficient RLHF variants (e.g., PERL) show this can substantially reduce training cost while maintaining alignment quality (Sidahmed et al., 2024).

Layer-selective tuning. Finally, we use the notion of *top-layer* tuning: only adapters inserted in the last L transformer blocks are updated. This choice is consistent with evidence that higher layers and FFN submodules often contain high-level attributes and alignment-relevant representations (Geva et al., 2021; Lai et al., 2024).

4 Methodology

We propose **PERSA** (Professor-Style Reinforcement-based Style Adaptation), a parameter efficient RLHF framework that aligns an LLM's *feedback style* to a target instructor while preserving the model's underlying problem-solving and language competence. PERSA instantiates the standard RLHF recipe, *supervised fine-tuning* $\rightarrow$ *reward modeling* $\rightarrow$ *policy optimization*, popularized by InstructGPT (Ouyang et al., 2022), but crucially restricts learning to *style-bearing* components in the *upper* transformer layers using low-rank adapters (Hu et al., 2022; Sidahmed et al., 2024). This design is motivated by growing evidence that higher layers and FFN submodules disproportionately encode high-level attributes such as discourse patterns, tone, and stylistic preferences (Geva et al., 2021; Lai et al., 2024). Figure 2 illustrates how PERSA transforms a generic prompt into instructor-style feedback by combining LoRA-based supervised fine-tuning with KL-regularized PPO (trained on professor data).

Problem Setup and Notation. Let π_θ denote a pretrained decoder-only transformer LLM (e.g., LLaMA/Gemma/GPT-style), parameterized by θ. Each training instance consists of a prompt x and a target feedback $y^\star$ written by the professor. The prompt x contains a programming problem statement and a student submission. The goal is to learn a policy π_θ that produces feedback y that is simultaneously: (i) *diagnostically correct* (identifies correctness/errors and provides actionable guidance) and (ii) *stylistically aligned* with the professor (tone, structure, politeness, specificity).

We use the demonstration and preference datasets defined in §3, namely $\mathcal{D}_{\text{SFT}}$ for professor-written feedback and $\mathcal{D}_{\text{pref}}$ for instructor-style preference comparisons.

Parameter-Efficient, Layer-Selective Updates. Rather than updating all parameters, we freeze the base model weights and learn low-rank adapter updates (LoRA) only in the *top* L transformer blocks (e.g., $L = 4$), focusing on attention projections and Feed-Forward Network projections. For a weight matrix $W \in \mathbb{R}^{d\times k}$ in a selected module, LoRA parameterizes, as detailed mentioned in equation 3. This yields a small trainable parameter set, θ_{LoRA} (on the order of $\sim 10^7$ parameters in our setting), while leaving the remaining parameters fixed, improving stability and reducing risk of catastrophic drift (Sidahmed et al., 2024). Our layer-selection choice is supported by analyses indicating that alignment- and style-related changes concentrate in later layers, often in the penultimate FFNs (Geva et al., 2021; Liang et al., 2024).

4.1 Supervised Fine-Tuning (SFT)

We first perform instruction tuning to initialize a policy that produces professor-like feedback under

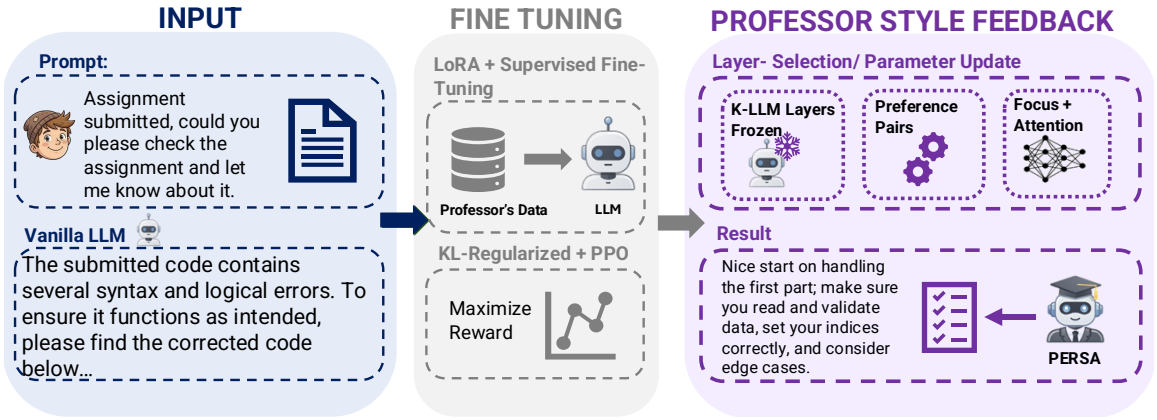

Figure 2: PERSA pipeline for professor-style feedback alignment via layer-selective RLHF.

maximum likelihood. The SFT objective is the standard auto-regressive negative log-likelihood, defined in equation 1. where only θ_{LoRA} is updated and all other parameters are frozen. This stage transfers the professor's feedback structure (e.g., diagnosis → correction → verification) and baseline tone into the model, but may not optimally capture preference-sensitive nuances (e.g., strictness, encouragement, emphasis on edge cases).

4.2 RLHF for Professor-Style Alignment

SFT imitates demonstrations but does not explicitly optimize for *human preference*. We therefore apply RLHF, consisting of reward modeling followed by PPO-based policy optimization, following established alignment practice (Ouyang et al., 2022).

4.2.1 Reward Modeling from Pairwise Preferences

We train a reward model $r_\phi(x, y) \in \mathbb{R}$ that scores candidate feedback y for prompt x. The reward model is a transformer initialized from a strong LM backbone with an added scalar head, trained on pairwise comparisons using the Bradley Terry formulation:

$$\Pr\Big(y^{(w)} \succ y^{(l)} \mid x\Big) = \sigma\Big(r_\phi(x, y^{(w)}) - r_\phi(x, y^{(l)})\Big), \tag{4}$$

where σ is the sigmoid function. Minimizing the negative log-likelihood yields:

$$\mathcal{L}_{\text{RM}}(\phi) = -\mathbb{E}_{(x,y^{(w)},y^{(l)})\sim\mathcal{D}_{\text{pref}}} \Big[\log \sigma\Big(r_\phi(x, y^{(w)}) - r_\phi(x, y^{(l)})\Big)\Big]. \tag{5}$$

Because preferred responses are professor-authored (or instructor-preferred), r_ϕ learns to jointly reflect correctness and stylistic fidelity, mirroring the RLHF setup in InstructGPT but with an education-specific preference signal (Ouyang et al., 2022).

4.2.2 Policy Optimization with PPO and KL Control

Given r_ϕ, we optimize the policy to maximize expected reward while constraining deviation from a reference policy (the SFT model) via a KL penalty. Let π_θ be the current policy and π_{ref} a frozen copy of the SFT policy. We optimize:

$$\max_{\theta_{\text{LoRA}}} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[r_\phi(x, y) - \beta\, \text{KL}(\pi_\theta(\cdot|x) \,\|\, \pi_{\text{ref}}(\cdot|x))\Big] \tag{6}$$

where $\beta > 0$ controls the strength of KL regularization, a key stabilizer in RLHF (Ouyang et al., 2022).

We solve Eq. (6) with PPO (Schulman et al., 2017). For token-level actions a_t under states s_t (prefix context), we define the importance ratio $\rho_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$ and advantage estimate $\hat{A}_t$. The clipped PPO objective is:

$$\mathcal{L}_{\text{PPO}}(\theta_{\text{LoRA}}) = -\mathbb{E}_t\Big[\min\big(\rho_t(\theta)\hat{A}_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big]. \tag{7}$$

with clipping threshold ϵ. We additionally include the KL control term (computed against π_{ref}) in the per-trajectory reward or as an auxiliary penalty, consistent with the constrained-RL perspective in RLHF (Ouyang et al., 2022). As in SFT, we update only θ_{LoRA} in the selected upper layers, yielding a parameter-efficient instantiation of RLHF akin to PERL (Sidahmed et al., 2024). This selective optimization reduces compute and helps preserve the pretrained model's broad capabilities.

Summary. PERSA proceeds as follows: (i) attach LoRA adapters to the top L layers, (ii) run SFT on $\mathcal{D}_{\text{SFT}}$ minimizing Eq. (1), (iii) train a reward model on $\mathcal{D}_{\text{pref}}$ minimizing Eq. (5), and (iv) run PPO to optimize Eq. (7) under the KL-regularized objective Eq. (6). The related steps are outlined in Alg. 1, Appendix A. This pipeline is architecture-agnostic for modern decoder-only LLMs and is especially suitable for educational personalization, where the goal is to preserve task competence while adapting communicative style.

5 Experimental Setup

5.1 Datasets

We evaluate PERSA on three code-feedback datasets: one instructor-authored APPS-style professor-feedback dataset and two public code-feedback benchmarks, PyFiXV and CodeReviewQA. We construct a standard 70/10/20 train/validation/test split with disjoint programs for PERSA training and evaluation.

(i) APPS Benchmark (professor-feedback dataset). To study realistic instructional personalization, we use a course-specific dataset of 200 instances, each consisting of an algorithmic programming problem, a student solution (either correct or deliberately perturbed), and *professor-written feedback* as the target response. The prompts are derived from APPS-style problems, while the feedback is authored by the instructor for the target course (Hendrycks et al., 2021).

(ii) Codeforces Syntax-Error Feedback (PyFiXV). We use the Codeforces subset of PyFiXV, which contains 240 Python programs with syntax errors (token length ≤ 500), collected from Codeforces and paired with expert-written fixes and explanations (Phung et al., 2023).

(iii) `Tomo-Melb/CodeReviewQA`. CodeReviewQA comprises 900 manually curated code-review instances spanning 9 programming languages. Each example includes a pre-review snippet (`old`), a reviewer comment (`review`), and the revised code (`new`). We cast this benchmark as a text-to-text task, mapping $(x = \{\texttt{old}, \texttt{review}\})$ to $y = \texttt{new}$ (Lin et al., 2025).

Appendix B.1 provides details on the demonstration and preference data, while Table 6 summarizes the data splits and corresponding leakage controls.

5.2 Model Evaluation Frameworks

We evaluate PERSA on two lightweight, open-weight instruction-tuned backbones, `google/gemma-2-2b-it` and `meta-llama/Llama-3.2-3B-Instruct`, within a model-agnostic RLHF framework comprising supervised fine-tuning, reward modeling, and PPO-based policy optimization with KL regularization. We compare against a diverse set of baselines, including the untuned **Base Model**, **SFT** (Parthasarathy et al., 2024), **InstructGPT-style RLHF** (Ouyang et al., 2022), and three offline preference-optimization methods, namely **DPO** (Wu et al., 2024b), **ORPO** (Hong et al., 2024), and **KTO** (Ethayarajh et al., 2024). This setup enables a controlled comparison of stylistic alignment, correctness preservation, and adaptation efficiency across both model families under a shared evaluation protocol (Appendix B.2).

5.3 Evaluation Metrics

We evaluate feedback generation along two complementary dimensions: *stylistic alignment* and *diagnostic fidelity*. To quantify how closely the generated feedback matches the instructor's communicative style, we use **Style Alignment Score (SAC)** (Fu et al., 2018), which estimates the probability that a response reflects the professor's stylistic characteristics. We further measure tone similarity using **Average Politeness Closeness (APC)** (Danescu-Niculescu-Mizil et al., 2013), which evaluates how closely the politeness level of the generated feedback aligns with the professor reference. Finally, we report **BLEU-4** (Papineni et al., 2002) to capture lexical similarity by measuring n-gram overlap between generated feedback and the professor's reference response. All together, they measure stylistic fidelity to the professor's feedback in terms of overall style alignment, politeness closeness, and lexical similarity, respectively.

Correctness Accuracy (CA) (Shen et al., 2023) assesses whether the generated feedback makes the correct diagnostic judgment about the student solution. In addition, **Preference Win Rate (PWR)** (Ouyang et al., 2022) measures preference-based superiority relative to competing methods, reflecting the alignment objective used in RLHF. Full metric definitions, formulations, and implementation details are provided in Appendix B.3. Training configurations are provided in Appendix B.4.

Table 1: Comparison of Llama-3 vs. Gemma-2 models across three datasets (APPS, PyFiXV, and CodeReviewQA). Higher values indicate better performance; all metrics are reported in percentages.

Dataset	Method	Llama-3 (%)					Gemma-2 (%)				
		SAC	APC	BLEU-4	CA	PWR	SAC	APC	BLEU-4	CA	PWR
APPS	Base	34.8	85.0	6.4	98.2	–	20.0	90.0	2.0	98.0	–
	SFT	82.0	86.0	80.0	100	86.2	85.0	95.0	70.0	100	90.0
	InstructGPT/RLHF	84.0	86.4	80.0	100	88.0	88.0	95.0	80.0	100	95.0
	DPO	94.0	87.8	94.2	100	89.8	86.0	95.0	78.0	100	94.0
	ORPO	95.6	89.0	95.0	100	**90.2**	89.0	95.0	80.0	100	98.0
	KTO	95.0	87.2	94.6	100	90.0	90.0	95.0	81.0	100	97.2
	PERSA	**96.2**	**92.1**	**95.8**	**100**	90.1	**99.0**	**95.0**	**98.0**	**100**	**98.0**
PyFiXV	Base	30.5	84.0	8.0	97.5	–	20.0	90.0	5.0	96.0	–
	SFT	79.0	85.5	76.0	99.5	84.5	80.0	94.0	60.0	100	85.0
	InstructGPT/RLHF	81.0	86.0	77.5	99.7	86.0	82.0	95.0	70.0	100	90.0
	DPO	92.0	87.0	92.5	99.8	88.0	81.0	95.0	68.0	100	88.0
	ORPO	93.5	88.0	93.2	99.8	88.6	83.0	95.0	72.0	100	98.0
	KTO	93.0	87.0	93.0	99.8	88.3	84.0	95.0	73.0	100	96.0
	PERSA	**94.5**	**91.0**	**94.0**	**99.9**	**89.0**	**98.0**	**95.0**	**98.0**	**100**	**98.2**
CodeReviewQA	Base	28.0	83.5	8.0	96.8	–	15.0	90.0	3.0	95.0	–
	SFT	78.0	85.0	75.0	99.2	83.5	80.0	94.0	65.0	100	85.0
	InstructGPT/RLHF	80.0	85.5	76.5	99.5	85.0	84.0	95.0	75.0	100	90.0
	DPO	91.0	86.8	92.0	99.7	87.0	82.0	95.0	73.0	100	89.0
	ORPO	92.2	87.5	92.7	99.7	87.5	86.0	95.0	76.0	100	98.0
	KTO	92.0	86.5	92.5	99.7	87.3	87.0	95.0	78.0	100	96.0
	PERSA	**93.8**	**90.5**	**93.5**	**99.8**	**88.0**	**98.0**	**95.0**	**98.0**	**100**	**98.2**

6 Results

6.1 Performance Comparison

Table 1 highlights consistent method-wise trends across all three datasets and both backbones. First, **Base** models achieve high correctness (CA≈96–98%) and moderate politeness (APC≈83–90%) but **very weak stylistic/lexical alignment** (low SAC and single-digit BLEU-4), indicating that pretraining alone does not recover professor-style feedback. Second, **SFT** provides the largest jump in alignment (e.g., SAC≈78–85%, BLEU-4≈60–80%) while keeping CA near-perfect, showing that demonstrations effectively teach the core feedback structure. Third, preference-based alignment further improves style: DPO/ORPO/KTO consistently outperform InstructGPT/RLHF in SAC and BLEU-4 with comparable CA, and yield higher win-rates (PWR), suggesting that direct preference optimization better captures subtle stylistic cues.

Figure 3 shows that PERSA delivers consistently stronger human preference alignment than SFT, RLHF, and DPO-style baselines on PyFiXV, with improvements that generalize across both Llama-3 and Gemma-2 backbones. Finally, PERSA achieves the strongest overall results on both models, delivering the highest SAC and BLEU-4 and the best PWR across datasets (with CA≈100%), demonstrating robust professor-style transfer without sacrificing diagnostic correctness.

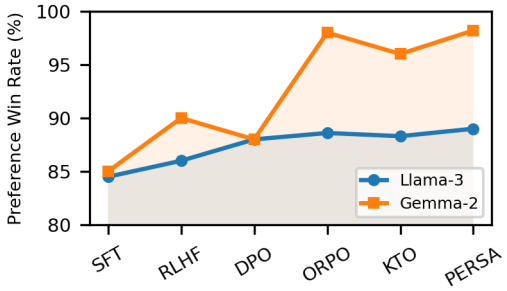

Figure 3: PWR comparison across alignment methods on the PyFiXV dataset for Llama-3 and Gemma-2.

6.2 Qualitative Analysis

We sample representative programming prompts from the APPS dataset and reproduce the corresponding feedback outputs. Table 2 mirrors the quantitative rankings in Table 1: methods with higher SAC/BLEU-4 (DPO/ORPO/KTO, and especially PERSA) produce feedback that is *more diagnosis-specific*, *actionable*, and *professor-structured* (diagnosis → fix → verification), whereas Base and standard RLHF remain comparatively generic, consistent with their lower style-

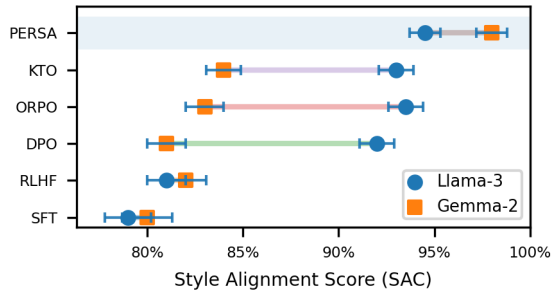

Figure 4: Dumbbell plot of SAC on PyFiXV across alignment methods, comparing Llama-3 and Gemma-2; connectors indicate the cross-model gap per method (higher is better).

Table 2: **Qualitative comparison (one prompt).** PERSA produces the most professor-like feedback: structured diagnosis → actionable fix → verification. Preference-optimization methods (ORPO/KTO/DPO) are typically more specific than generic RLHF, while Base tends to be vague.

Method	Qualitative feedback snippet (same prompt)
Prompt (APPS-style). Student computes `x=a.index(1), y=a.index(n)` before `a=list(read())`, causing a runtime error / wrong output.	
Base	"Try checking input handling and edge cases. Consider swapping min/max positions, then rerun tests."
SFT	"You are calling `a.index()` before initializing `a`. Move `a=list(read())` above the `index` calls, then recompute `x,y`."
InstructGPT	"Good start. The issue is order of operations: read the array first, then find indices and apply the swap. Verify with sorted/reversed cases."
DPO	"Root cause: `a` is used before assignment. Fix by reading input into `a` first, then compute `x=a.index(1)` and `y=a.index(n)`. Add tests for $n{=}1$ and already-sorted."
ORPO	"Bug: `a.index()` runs before `a=list(read())`. **Initialize `a` first**, then compute `x,y` and perform one swap maximizing $\|x{-}y\|$. **Re-run edge cases** (sorted, reversed, single displacement)."
KTO	"You are using `a` before it is populated. Read input into `a` first, then compute indices and swap. Avoid overclaiming; confirm correctness with boundary cases and failing tests."
PERSA	**"Good progress on locating 1 and n. The failure arises from using `a.index()` before `a` is populated. Please first read input with `a=list(read())`; then recompute `x=a.index(1), y=a.index(n)` and apply one swap maximizing $\|x{-}y\|$. Finally, rerun edge cases (already-sorted, reversed, $n{=}1$) to confirm."**

alignment scores and preference win-rates. As shown in Figure 4, PERSA achieves the highest SAC for both backbones while maintaining consistent gains over SFT and preference-optimization baselines, indicating strong and portable style alignment across models.

6.3 Ablation Study

Table 3 quantifies how each training component translates into measurable style gains. Moving from Base to SFT converts a largely generic model (SAC 14.0%, BLEU-4 1.5) into a professor-like feedback generator (SAC 82.0%, BLEU-4 64.7) while reaching perfect diagnostic correctness (CA=100%). However, PPO alone does not

Table 3: PERSA ablation on Llama-3 and APPS dataset. Metrics are in % (higher is better); PWR is win-rate vs. Base.

PERSA Ablation	SAC	APC	BLEU-4	CA	PWR
Base (no adapt.)	14.0	90.0	1.5	98.0	–
SFT only	82.0	91.6	64.7	100	86.0
PPO only (no SFT)	60.0	91.2	40.0	98.0	84.0
SFT+PPO (full-param)	92.0	92.0	92.0	100	88.0
SFT+PPO (all-layer LoRA)	96.0	92.0	94.7	100	88.6
SFT+PPO (top-2 LoRA)	94.0	92.0	90.0	100	90.0
SFT+PPO (top-4 LoRA) [PERSA]	**96.2**	**92.1**	**95.8**	**100**	**90.1**

recover the same instructor voice (SAC 60.0%, BLEU-4 40.0) and even leaves a small correctness gap (CA=98.0%), indicating that preference optimization needs an SFT "anchor" to be effective. When PPO is applied after SFT, alignment improves sharply: full-parameter PPO attains 92.0% SAC and 92.0 BLEU-4 with CA=100%, showing that RLHF mainly refines the remaining stylistic mismatch rather than content, also mentioned in Figure 4. Constraining updates via adapters is even stronger all-layer LoRA pushes SAC to 96.0% and BLEU-4 to 94.7 without sacrificing correctness suggesting that parameter-efficient updates add expressive capacity while limiting drift. Finally, the best numbers come from upper-layer targeting: top-2 LoRA already reaches high alignment (SAC 94.0%, BLEU-4 90.0) and strong preference (PWR=90.0%), while top-4 LoRA (PERSA) achieves peak alignment (SAC=96.2%, BLEU-4=95.8) and the highest win-rate (PWR=90.1%), consistent with style information being concentrated in late transformer layers.

6.4 Human Evaluation Study

To complement automated metrics, we conducted two light weight human evaluations using anonymized Forms: (i) an *Instructor* blind A/B preference + rubric study and (ii) a *Student* perception study. In the instructor study, participants reviewed programming prompts with a student attempt and compared two anonymous feedback responses (Feedback A vs. Feedback B), selecting an overall preference and rating multiple pedagogical criteria (e.g., technical correctness, authenticity, helpfulness, and trust-oriented tone) using a short rubric (Feedback A (very good), Feedback A (good), Neither (neutral), Both (equal), Feedback B (good), Feedback B (very good)). The form explicitly instructs instructors to judge pedagogical quality rather than minor grammatical issues and reports an estimated completion time of 8–12 minutes. In the student study, each participant reviewed

Table 4: Student survey average ratings (on a 1–5 Likert scale) across 20 respondents and 5 examples.

Statement	Rating (stdev)
The feedback was clear.	4.34 (1.05)
The feedback was helpful.	4.37 (1.04)
I trust this feedback.	4.31 (0.97)
It includes a full solution/code I could copy directly.	2.55 (1.49)
I know what the issue is after reading the feedback.	4.27 (1.07)
It sounds like a real instructor.	3.87 (1.34)

Table 5: Instructor survey across 12 respondents (blind A/B). Percentage of cases where feedback A (PERSA) was preferred over Feedback B (vanilla LLM); tie aggregates "Both"/"Neither" responses.

Statement	A (%)	B (%)	Tie (%)
Overall preference	83.6	1.8	14.6
Helpfulness / actionability	85.5	1.8	12.7
Authenticity (human instructor)	74.5	1.8	23.7
Trust-oriented tone	87.3	3.6	9.1
Technical correctness	61.8	5.5	32.7
Gives too much solution	27.3	27.3	45.4

five feedback instances and reported an evaluation of different features of the feedback on a Likert (1–5) scale.

Human-study interpretation. Tables 4 and 5 summarize our blind human evaluation comparing **PERSA** (Feedback A) to a vanilla LLM (Feedback B) across diverse respondents, including university instructors with varying teaching experience and students spanning beginner to advanced programming backgrounds. We used short, consistent rubrics that asked participants to judge clarity, helpfulness / action-ability, trust, and instructor-likeness (rather than minor grammar). We observe converging evidence that PERSA's professor-style alignment improves perceived feedback quality beyond the baseline. Students report high ratings for clarity, helpfulness, trust, and understanding (4.27–4.37/5), while the lower score on "includes a full solution" (2.55) suggests the feedback typically supports revision without excessive solution dumping. Instructors similarly prefer PERSA (83.6%) and rate it higher on helpfulness/actionability (85.5%) and trust-oriented tone (87.3%), with substantially stronger perceived authenticity (74.5%) than the vanilla LLM.

To complement our automated evaluation, we conducted a human study involving 32 participants (comprising faculty and students) across 8 universities. This study compares PERSA against a vanilla LLM baseline to evaluate perceived pedagogical quality and alignment with instructor-like behavior. As shown in Figure 5, respondents consistently preferred PERSA over the vanilla baseline in both overall preference and human-like, instructor-style feedback, providing direct human-centered evidence that PERSA produces responses that are more authentic, supportive, and pedagogically aligned. Details of the study design are provided in Appendix C.

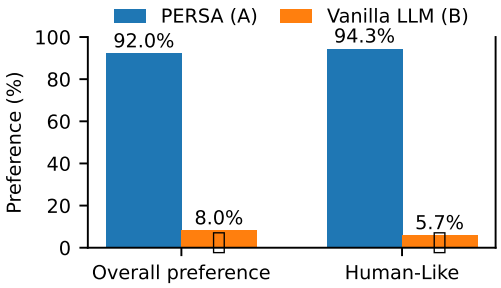

Figure 5: Human-study outcomes comparing PERSA against a vanilla LLM.

7 Conclusion

We presented **PERSA**, a professor-style reinforcement learning framework that adapts LLM-generated programming feedback to an instructor's communicative voice while maintaining diagnostic correctness. PERSA follows an RLHF recipe, supervised fine-tuning, preference-based reward modeling, and PPO optimization, but makes adaptation practical and controllable by restricting learning to style-relevant parameters through *layer-selective*, parameter-efficient LoRA updates concentrated in the upper transformer layers. Across datasets and model backbones, PERSA achieves the strongest overall alignment, reaching near-ceiling style and lexical similarity (e.g., SAC/BLEU) without degrading correctness accuracy, and producing feedback that is consistently more structured, specific, and pedagogically actionable in qualitative analyses. Taken together, our results suggest that "instructor voice" behaves as a largely separable and optimizable attribute that can be injected via targeted adaptation rather than full-model retraining, enabling more trustworthy, authentic, and deployable AI feedback systems for education.

Limitations

PERSA is evaluated on a limited set of instructors, programming tasks, and feedback distributions, so its gains in stylistic alignment may not directly transfer to new courses, grading philosophies, or broader educational settings without additional preference data and recalibration. Moreover, because the reward model is learned from a specific instructor cohort and prompting setup, it may capture dataset-specific biases or stylistic artifacts, and, like RLHF more broadly, it remains vulnerable to reward misclassifications and over-optimization. Detailed discussion, limitations, and future directions are provided in Appendix D.

Ethical Considerations

We carefully considered the ethical responsibilities associated with this work, particularly the human evaluation component. The study was conducted in accordance with standard institutional and legal expectations for voluntary human participation, including anonymized response collection, minimal-risk survey design, and analysis in aggregate form without storing personally identifiable information.

References

Bashaer Alsafari, Eric Atwell, Aisha Walker, and Martin Callaghan. 2024. Towards effective teaching assistants: From intent-based chatbots to llm-powered teaching assistants. *Natural Language Processing Journal*, 8:100101.

Pronaya Bhattacharya, Vivek Kumar Prasad, Ashwin Verma, Deepak Gupta, Assadaporn Sapsomboon, Wattana Viriyasitavat, and Gaurav Dhiman. 2024. Demystifying chatgpt: An in-depth survey of openai's robust large language models. *Archives of Computational Methods in Engineering*, 31(8):4557–4600.

David Carless. 2012. Trust and its role in facilitating dialogic feedback. In *Feedback in higher and professional education*, pages 90–103. Routledge.

Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. 2025. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *ACM Computing Surveys*, 58(2):1–37.

Dongwook Choi and Eunseok Lee. 2025. Automated feedback generation for programming assignments through diversification. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 230–241. IEEE.

Wei Dai, Yi-Shan Tsai, Jionghao Lin, Ahmad Aldino, Hua Jin, Tongguang Li, Dragan Gašević, and Guanliang Chen. 2024. Assessing the proficiency of large language models in automatic feedback generation: An evaluation study. *Computers and Education: Artificial Intelligence*, 7:100299.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 250–259.

Dorottya Demszky, Jing Liu, Heather C Hill, Dan Jurafsky, and Chris Piech. 2024. Can automated feedback improve teachers' uptake of student ideas? evidence from a randomized controlled trial in a large-scale online course. *Educational Evaluation and Policy Analysis*, 46(3):483–505.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. 2024. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Iria Estévez-Ayres, Patricia Callejo, Miguel Ángel Hombrados-Herrera, Carlos Alario-Hoyos, and Carlos Delgado Kloos. 2024. Evaluation of LLM tools for feedback generation in a course on concurrent programming. *International Journal of Artificial Intelligence in Education*, 35(2).

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Richard M Felder, Linda K Silverman, and 1 others. 1988. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Hagit Gabbay and Anat Cohen. 2024. Combining LLM-generated and test-based feedback in a MOOC for programming. In *Proceedings of the 11th ACM Conference on Learning @ Scale (L@S 2024)*, pages 177–187. ACM.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5484–5495. Association for Computational Linguistics.

Luke Gusukuma, Austin Cory Bart, and Dennis Kafura. 2020. Pedal: An infrastructure for automated feedback systems. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. ACM.

Jieun Han, Haneul Yoo, Junho Myung, Minsun Kim, Hyunseung Lim, Yoonsu Kim, Tak Yeon Lee, Hwajung Hong, Juho Kim, So-Yeon Ahn, and 1 others. 2023. Llm-as-a-tutor in efl writing education: Focusing on evaluation of student-llm interaction. *arXiv preprint arXiv:2310.05191*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and 1 others. 2021. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *arXiv preprint arXiv:2403.07691*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Uday Kamath, Kevin Keenan, Garrett Somers, and Sarah Sorenson. 2024. Tuning for llm alignment. In *Large Language Models: A Deep Dive: Bridging Theory and Practice*, pages 177–218. Springer.

Hieke Keuning, Johan T. Jeuring, and Bastiaan Heeren. 2018. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education*, 19(1):1–43.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Fajri Koto, Jey Han Lau, and Timothy Baldwin. 2022. Neuron-level interpretation of deep nlp models: A survey. *Transactions of the Association for Computational Linguistics*, 10:1285–1303. Surveys methods for interpreting style-related signals in transformer models, relevant for identifying style-specific neurons in higher layers.

Wen Lai, Viktor Hangya, and Alexander Fraser. 2024. Style-specific neurons for steering llms in text style transfer. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13427–13443.

Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Yi Ding, Xinyu Yang, Kailas Vodrahalli, Siyu He, Daniel Scott Smith, Yian Yin, and 1 others. 2024. Can large language models provide useful feedback on research papers? a large-scale empirical analysis. *NEJM AI*, 1(8):AIoa2400196.

Hong-Yi Lin, Chunhua Liu, Haoyu Gao, Patanamon Thongtanunam, and Christoph Treude. 2025. Codereviewqa: The code review comprehension assessment for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9138–9166.

Euan D Lindsay, Mike Zhang, Aditya Johri, and Johannes Bjerva. 2025. The responsible development of automated student feedback with generative ai. In *2025 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–10. IEEE.

John Loughran. 2019. Pedagogical reasoning: The foundation of the professional knowledge of teaching. *Teachers and Teaching*, 25(5):523–535.

Yun Luo, Zhen Yang, Fandong Meng, Yuchen Li, Jie Zhou, and Yue Zhang. 2023. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*.

Andrea Matarazzo and Riccardo Torlone. 2025. A survey on large language models with some insights on their capabilities and limitations. *arXiv preprint arXiv:2501.04040*.

Milad Moradi, Ke Yan, David Colwell, Matthias Samwald, and Rhona Asgari. 2024. Exploring the landscape of large language models: Foundations, techniques, and challenges.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Sinan Ozdemir. 2023. *Quick start guide to large language models: strategies and best practices for using ChatGPT and other LLMs*. Addison-Wesley Professional.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

Tung Phung, José Cambronero, Sumit Gulwani, Tobias Kohn, Rupak Majumdar, Adish Singla, and Gustavo Soares. 2023. Generating high-precision feedback for programming syntax errors using large language models. *arXiv preprint arXiv:2302.04662*.

Mahefa Abel Razafinirina, William Germain Dimbisoa, and Thomas Mahatody. 2024. Pedagogical alignment of large language models (LLM) for personalized learning: a survey, trends and challenges. *Journal of Intelligent Learning Systems and Applications*, 16(4):448–480.

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

Tianhao Shen, Chenyang Zhou, Tianyu Wu, Canwen Wang, Xuming Wang, and Steven C. H. Hoi. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*. Discusses challenges in evaluating LLMs, including metrics for content correctness and alignment, relevant for Correctness Accuracy (CA) in feedback evaluation.

Guangyuan Shi, Zexin Lu, Xiaoyu Dong, Wenlong Zhang, Xuanyu Zhang, Yujie Feng, and Xiao-Ming Wu. 2024. Understanding layer significance in llm alignment. *arXiv preprint arXiv:2410.17875*.

Hakim Sidahmed, Samrat Phatale, Alex Hutcheson, Zhuonan Lin, Zhang Chen, Zac Yu, Jarvis Jin, Simral Chaudhary, Roman Komarytsia, Christiane Ahlheim, and 1 others. 2024. Parameter efficient reinforcement learning from human feedback. *arXiv preprint arXiv:2403.10704*.

Sinan Sonlu, Bennie Bendiksen, Funda Durupinar, and Uğur Güdükbay. 2024. The effects of embodiment and personality expression on learning in llm-based educational agents. *arXiv preprint arXiv:2407.10993*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, and 1 others. 2023a. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023b. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Di Wu, Meng Chen, Xu Chen, and Xing Liu. 2024a. Analyzing k-12 ai education: A large language model study of classroom instruction on learning theories, pedagogy, tools, and ai literacy. *Computers and Education: Artificial Intelligence*, 7:100295.

Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. 2024b. Dpo: Direct preference optimization with dynamic. *Advances in Neural Information Processing Systems*, 37:129944–129966.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, and 1 others. 2024. Sorry-bench: Systematically evaluating large language model safety refusal. *arXiv preprint arXiv:2406.14598*.

Yuzi Yan, Xingzhou Lou, Jialian Li, Yiping Zhang, Jian Xie, Chao Yu, Yu Wang, Dong Yan, and Yuan Shen. 2024. Reward-robust rlhf in llms. *arXiv preprint arXiv:2409.15360*.

Zhe Victor Zhang and Ken Hyland. 2022. Fostering student engagement with feedback: An integrated approach. *Assessing Writing*, 51:100586.

Appendix

A Pseudo Code

Algorithm 1: PERSA: Layer-Selective LoRA + RLHF

Input : Base LLM π_θ; demos $\mathcal{D}_{\text{SFT}}$; prefs $\mathcal{D}_{\text{pref}}$; top layers L; LoRA rank r; KL β; PPO steps T

Output : Style-aligned policy $\pi_{\theta'}$

Freeze θ; insert LoRA (rank r) in top L layers; trainable params θ_{LoRA};

SFT:;

$\theta_{\text{LoRA}} \leftarrow \arg\min_{\theta_{\text{LoRA}}} \mathbb{E}_{(x,y^\star)\sim\mathcal{D}_{\text{SFT}}}\left[-\sum_t \log \pi_\theta(y_t^\star \mid x, y_{<t}^\star)\right]$;

Set reference $\pi_{\text{ref}} \leftarrow \pi_\theta$ (frozen);

RM:;

Train reward r_ϕ on $\mathcal{D}_{\text{pref}}$ with $\mathcal{L}_{\text{RM}} = -\mathbb{E}\left[\log \sigma(r_\phi(x, y^{(w)}) - r_\phi(x, y^{(l)}))\right]$;

PPO:;

for $t = 1$ **to** T **do**

  Sample x; generate $y \sim \pi_\theta(\cdot|x)$; compute $\tilde{R} = r_\phi(x, y) - \beta\,\mathrm{KL}(\pi_\theta \| \pi_{\text{ref}})$;

  Update θ_{LoRA} by PPO using reward $\tilde{R}$ (clipped ratio);

return $\pi_{\theta'}$ (base + trained LoRA);

B Detailed Experimental Settings

B.1 Data Splits & Leakage Controls.

To prevent train-test contamination and inflated SAC/BLEU scores from near-duplicate examples, we employ leakage-aware splits and explicit overlap audits. In addition to a standard *instance-level* split, we evaluate on a stricter *New-Problems* split that withholds entire problem IDs from training. Before splitting, we remove exact duplicates using hashes over {problem, solution, reference feedback} and filter near-duplicates via MinHash/Jaccard similarity on problem text and reference feedback. After splitting, we compute each test example's maximum similarity to the training set and verify that all remain below a conservative threshold. To further reduce dependence on surface-level overlap, we also report paraphrase-robust SAC and diversity statistics. Table 6 summarizes these controls.

Preference data. The same three datasets are used to construct both D_{SFT} and D_{pref}. For D_{SFT}, each example is a prompt-reference pair $(x, y^\star)$, where $y^\star$ is the professor or expert-written feedback. For D_{pref}, we form pairwise comparisons $(x, y^{(w)}, y^{(l)})$ using the same training prompts: the preferred response $y^{(w)}$ is the professor/expert feedback, and the rejected response $y^{(l)}$ is feedback generated by the vanilla backbone or a weaker baseline under the same prompt. Preference pairs are created only from the training split to avoid train-test leakage.

Table 6: Leakage-aware splits & overlap audits.

Audit / Split	Std.	New-Problems
Split key	inst.	prob. ID
Exact dups rm. (%) ↓	1.2	1.2
Near-dups rm. (%) ↓	6.5	6.5
Max train–test sim. (problem) ↓	0.23	0.19
Max train–test sim. (ref. fb) ↓	0.18	0.15
# test items >0.8 (Jaccard) ↓	0	0
Paraphrase SAC agr. (%) ↑	94.0	91.5
BERTScore (F1) ↑	0.92	0.90
Self-BLEU ↓	0.54	0.50
Distinct-2 ↑	0.21	0.24

Table 7: Method footprint. PERSA retains an RLHF pipeline but updates only Top-L LoRA adapters (layer-selective PEFT), improving efficiency and style fidelity.

Method	Obj.	RM	On-pol.	Trainable	Scope	Pros
Base	–	N	N	0%	Frozen	No cost
SFT	SFT	N	N	Full	All	Stable, simple
RLHF	PPO	Y	Y	Full	All	Strong align; costly
DPO	DPO	N	N	Full	All	Offline, stable
ORPO	ORPO	N	N	Full	All	Efficient prefs
KTO	KTO	N	N	Full	All	Robust prefs
PERSA	**SFT+PPO**	**Y**	**Y**	**Top-L LoRA**	**Top-L**	**Style + low drift**

B.2 Model Evaluation Framework

We use lightweight, open-weight instruction-tuned LLMs as backbones so that PERSA (SFT + reward modeling + PPO) can run in a single-GPU setting. **Vanilla LLM baseline.** Unless otherwise stated, "vanilla LLM" refers to the instruction-tuned backbone used without task-specific fine-tuning or preference optimization. In our experiments, this corresponds to `google/gemma-2-2b-it` or `meta-llama/Llama-3.2-3B-Instruct`, evaluated with the same prompts and decoding configuration as the adapted models.

Primary backbone: Gemma 2 2B Instruct. We use `google /gemma-2-2b-it` as the main policy model due to its strong instruction-following behavior at a practical size, enabling parameter-efficient RLHF in constrained compute (Team et al., 2024). **Secondary backbone: Llama 3.2 3B Instruct.** We additionally evaluate `meta-llama/Llama-3.2-3B-Instruct` to test

cross-family robustness (Meta vs. Google) while keeping the model small enough for Colab-scale tuning (Dubey et al., 2024). **General recipe (model-agnostic).** All experiments treat the LLM as a conditional policy $\pi_\theta(y \mid x)$ over feedback tokens; we apply (1) supervised fine-tuning on instructor demonstrations, (2) reward modeling from pairwise preferences, and (3) PPO-based policy optimization with a KL constraint to preserve base knowledge. This structure is compatible with any decoder-only LLM that supports PEFT adapters (Ouyang et al., 2022; Schulman et al., 2017).

Methods Compared: We compare PERSA against strong baselines spanning supervised imitation and preference-based alignment to assess their ability to generate professor-like programming feedback while preserving correctness. (1) **Base Model** denotes the instruction-tuned backbone (Gemma-2-2B-IT or Llama-3.2-3B-Instruct) used *without* any task-specific adaptation. (2) **Supervised Fine-Tuning (SFT)** fine-tunes the backbone on professor-authored demonstrations using a maximum-likelihood objective. (3) **InstructGPT-style RLHF** follows the standard RLHF recipe of reward modeling from preferences and policy optimization with PPO and KL control (Ouyang et al., 2022). To reflect recent state-of-the-art alternatives to PPO-based RLHF, we additionally include three *offline* preference optimization methods trained on the same professor preference pairs: (4) **DPO** (Direct Preference Optimization), which directly optimizes the policy from preference comparisons without an explicit reward model (Wu et al., 2024b), (5) **ORPO** (Odds Ratio Preference Optimization), which integrates preference signals into an SFT-like objective (Hong et al., 2024), and (6) **KTO** (Kahneman–Tversky Optimization), a robust preference based objective inspired by prospect theory (Ethayarajh et al., 2024). Finally, (7) **PERSA** is our proposed framework that combines SFT initialization with *layer-selective, parameter-efficient* preference alignment (LoRA on upper layers) to target instructor style while minimizing drift in core capabilities. **Table 7 columns.** We report each method's optimization objective (*Obj.*), whether it requires an explicit reward model (*RM*) and on-policy rollouts (*On-pol.*), the fraction of trainable parameters (*Trainable*), and update scope across layers (*Scope*), highlighting that offline preference methods (DPO/ORPO/KTO) train full weights without RM/rollouts, while **PERSA** follows an RLHF pipeline (RM+on-policy) but updates only Top-L LoRA adapters for efficient, style-focused alignment. All approaches are evaluated under the same prompts, data splits, and decoding settings, enabling a controlled comparison of stylistic alignment and content fidelity.

B.3 Evaluation Metric

We evaluate PERSA on two complementary dimensions: *style alignment* (how feedback is phrased) and *diagnostic fidelity* (whether feedback is correct). All metrics are reported on the held-out test set as means.

Style Alignment Score (SAC). We train a calibrated binary style classifier (professor vs. non-professor feedback) and compute

$$\mathrm{SAC} = \frac{1}{N}\sum_{i=1}^{N} C(\hat{y}_i), \tag{8}$$

where $C(\hat{y}_i) \in [0,1]$ is the posterior for the `professor` label; higher is better (Fu et al., 2018).

Average Politeness Closeness (APC). Using a politeness scorer $p(\cdot) \in [0,1]$ (Danescu-Niculescu-Mizil et al., 2013), we measure tone closeness to the professor reference:

$$\mathrm{APC} = \frac{1}{N}\sum_{i=1}^{N} \Big(1 - \big|p(\hat{y}_i) - p(y_i^\star)\big|\Big). \tag{9}$$

BLEU-4. We report corpus BLEU-4 (with smoothing) to quantify n-gram overlap in phrasing with the professor feedback (Papineni et al., 2002).

Correctness Accuracy (CA). CA measures whether the feedback correctly identifies solution correctness. Let z_i be the unit-test label and $\hat{z}_i$ the judgment extracted from $\hat{y}_i$; then

$$\mathrm{CA} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[\hat{z}_i = z_i], \tag{10}$$

with higher indicating more reliable feedback (Shen et al., 2023). We interpret CA as a preservation check rather than the main source of improvement.

Preference Win Rate (PWR). To reflect RLHF objectives, we report the fraction of prompts where PERSA is preferred over a baseline under the same rater:

$$\mathrm{PWR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\Big[r_\phi(x_i, \hat{y}_i^{\mathrm{PERSA}}) > r_\phi(x_i, \hat{y}_i^{\mathrm{BASE}})\Big]. \tag{11}$$

a standard RLHF evaluation signal (Ouyang et al., 2022).

Metric reliability. We treat SAC and APC as automatic proxies rather than definitive measures of pedagogical quality. To check reliability, we calibrate the SAC classifier on held-out professor/non-professor feedback. For APC, we report correlation with human judgments of tone/trust-oriented feedback. These checks are intended to ensure that automatic metrics track human-perceived style dimensions rather than only surface lexical overlap.

B.4 Configurations

We summarize key hyperparameter for reproducible local-scale training; values below are defaults and can be scaled with model size.

Input formatting. We serialize each example as: *Problem* + *Student code* + optional *tests/constraints* $\rightarrow$ *Feedback*. We cap the input length (e.g., 1 2k tokens) and truncate from the left (oldest context) when needed.

Parameter-efficient tuning (LoRA/QLoRA). We inject LoRA adapters into attention and MLP projections (e.g., `q_proj`, `k_proj`, `v_proj`,`o_proj`, `up_proj`, `down_proj`, `gate_proj`) and update only the *top L transformer blocks* (layer-selective adaptation). Typical settings: rank $r \in \{8, 16\}$, $\alpha \in \{16, 32\}$, dropout $p = 0.05$. For coding environments, we use 4-bit NF4 quantization with double-quantization when available (QLoRA). (Hu et al., 2022; Dettmers et al., 2023)

SFT. Optimizer: AdamW; learning rate 1e-5 2e-5; epochs 3 10 (smaller for CodeReviewQA, larger for the 200-example instructor set); effective batch size via gradient accumulation; warmup 3 5%; weight decay 0.0 0.1; label smoothing optional (0.0 0.1).

Reward model (RM). We train a reward model $r_\phi(x, y)$ on preference pairs with a Bradley Terry / logistic loss; batch size 8 32; 1 3 epochs; early stopping on validation preference accuracy (Ouyang et al., 2022).

PPO (RLHF). We optimize the KL-regularized objective with PPO using: rollout batch 32 128 prompts, 1-4 PPO epochs per iteration, clipping $\epsilon = 0.2$, KL coefficient $\beta \in [0.005, 0.05]$, and a reference policy π_{ref} set to the post-SFT model. We keep generation short for feedback (e.g., max new tokens 128 256) to stabilize advantage estimates. (Schulman et al., 2017; Ouyang et al., 2022)

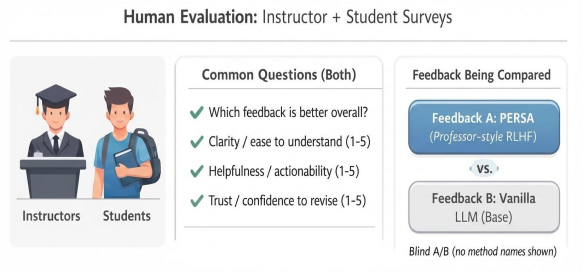

Figure 6: Overview of the human evaluation design used for instructor and student studies.

Compute and reproducibility. All runs fix random seeds, log training curves, and report mean$\pm$std over 3 seeds when feasible. On GPUs (A100), Gemma-2-2B with 4-bit LoRA typically fits in memory with moderate batch sizes; Llama-3.2-3B may require smaller rollouts or more accumulation.

C Human Study Design

Figure 6 summarizes our human evaluation protocol, showing the participating groups (instructors and students), the shared evaluation criteria (overall preference, clarity, helpfulness, and trust), and the blind A/B comparison between PERSA-generated feedback and vanilla LLM feedback used in both surveys.

Our instructor student **human evaluation section C**, provides direct evidence that PERSA's RLHF objective translates into perceived pedagogical gains participants overwhelmingly preferred the PERSA outputs over a vanilla LLM thereby validating that the learned preference signal improves feedback quality beyond automated style metrics. **Figure 5** states that across instructor and student respondents, PERSA is preferred over the vanilla LLM (92% vs. 8%) and is judged to better match a human-instructor feedback style (94.3%), providing human-centered validation of PERSA's RLHF-based style alignment.

D Discussion

PERSA demonstrates that *personalized pedagogical alignment* can be achieved by tuning *how* feedback is delivered (tone, structure, and phrasing) without degrading *what* is delivered (diagnostic correctness). Across three datasets and two model families, we observe a consistent separation between *style* and *substance*: base instruction-tuned models retain high correctness (CA) but exhibit weak professor-style fidelity (low SAC/BLEU),

while preference-based methods systematically improve stylistic alignment with minimal impact on CA. Crucially, PERSA achieves the strongest overall alignment (near-ceiling SAC/BLEU with CA$\approx$100%), indicating that instructor voice is an optimizable target distinct from generic "helpfulness" alignment.

Why each component matters. PERSA's performance is not due to a single stage, but to the complementary roles of its components. **SFT** provides a stable initialization that learns the instructor's canonical feedback template (e.g., encouragement $\rightarrow$ diagnosis $\rightarrow$ fix $\rightarrow$ verification), yielding the largest first jump in SAC/BLEU and perfect CA, but it does not fully capture nuanced preferences such as strictness, specificity, and emphasis on edge cases. **Reward modeling** operationalizes these nuanced stylistic and pedagogical preferences from pairwise comparisons, producing a scalar signal that encodes instructor-specific judgments beyond likelihood matching. **PPO with KL control** then refines the policy to maximize preference reward while preventing broad drift from the SFT reference, which is essential for preserving the backbone's general coding competence and avoiding capability regressions.

Why layer-selective LoRA is central. A key insight behind PERSA is that updating *all* parameters is neither necessary nor desirable for style personalization. Motivated by evidence that late transformer layers and FFN submodules disproportionately encode high-level attributes (discourse patterns, tone, and stylistic signatures), PERSA restricts trainable parameters to LoRA adapters in the top layers. This yields two benefits: (i) **efficiency** only a small fraction of parameters (e.g., $\sim$30M in the last 4 layers) need to move to achieve large stylistic gains; and (ii) **stability** constraining updates reduces catastrophic drift and preserves correctness, which is critical in educational feedback where factual errors can harm learning. The ablations support this design: PPO without SFT is sample-inefficient for learning instructor voice, and broader updates (full-parameter PPO or all-layer LoRA) improve style but underperform targeted top-layer adaptation, implying that "style-bearing capacity" is concentrated in late layers rather than uniformly distributed.

Relevance and broader implications. PERSA's results suggest a practical path to **course-level personalization** of AI tutors: rather than training new models per instructor, one can adapt a compact backbone with a small set of instructor-specific adapters and a preference signal, enabling rapid deployment and easy swapping of styles across courses. This is particularly relevant for educational settings where instructors have distinct grading philosophies (e.g., emphasis on readability vs. correctness vs. edge cases) and where student trust depends on perceived authenticity. Beyond education, the same principle generalizes to domains requiring **persona consistent communication** (e.g., customer support, clinical explanations, legal drafting) where maintaining core competence while matching a specific communication style is essential. Finally, the strong agreement between quantitative alignment metrics and qualitative examples indicates that optimizing for instructor preference can yield feedback that is not only stylistically similar but also more actionable and pedagogically structured, supporting the use of preference based objectives as a reliable mechanism for fine-grained style alignment.

What PERSA adds beyond standard RLHF. PERSA does not claim a new reinforcement learning algorithm; rather, it operationalizes RLHF for a narrower educational alignment problem: instructor-specific feedback style. Its novelty lies in treating professor voice as a distinct optimizable attribute, combining instructor-authored demonstrations, preference modeling, and layer-selective LoRA updates to shift feedback tone, structure, and specificity while limiting parameter drift. This framing differs from generic RLHF, which optimizes broad helpfulness, and from standard SFT, which imitates demonstrations without explicitly optimizing instructor-preference signals.

Limitations. Our evaluation covers a small number of instructors, student feedback distributions, and programming tasks; consequently, the observed style improvements may not directly transfer to new instructors, grading policies, courses, or broader educational settings without additional preference data and validation. In addition, because PERSA is trained with a fixed prompting/template and a specific instructor cohort, the reward model may capture prompt- or dataset-specific artifacts, which can reduce robustness when moving to different assignment formats, programming languages, or institutional contexts unless recalibrated. Finally, as with RLHF more broadly, a learned reward can reflect annotator bias and remains vulnerable to misspecification or over-optimization even with KL regularization, motivating careful monitoring

and human-in-the-loop oversight prior to classroom deployment (Ouyang et al., 2022; Lindsay et al., 2025).

While our human study partially validates the automatic style metrics, we do not yet include an LLM-as-a-judge evaluation for holistic pedagogical style. Future work will compare SAC/APC with expert-calibrated LLM judges that assess authenticity, encouragement, specificity, and instructional tone.

Future Work. We will extend PERSA to multi-instructor and multi-course settings with longitudinal classroom studies, and will explore stronger preference collection and safety controls (e.g., uncertainty-aware rewards and calibration) to ensure style-personalized feedback remains accurate, fair, and robust in real deployments.

Educational outcomes beyond style. to complement style/correctness metrics (SAC /APC/BLEU/CA/PWR), we will evaluate *learning-relevant* outcomes: (i) revision success by measuring whether students can fix the identified issues on a resubmission task (pass@1 / unit-test pass rate improvement), and (ii) error recurrence by tracking whether the same misconception/bug type reappears in a follow-up attempt. In addition, we will assess **action-ability and rubric alignment** using a validated feedback rubric, rated by instructors/TA annotators, rather than preference alone.