CONTRASTIVE SELF-DISTILLATION FROM USER FEEDBACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models deployed in production generate a large volume of real interactions with users. Users rarely provide explicit scores for responses, but they often correct errors, clarify requirements, or express dissatisfaction through follow-up messages, making subsequent user messages a cheap yet rich learning signal. A class of self-distillation methods, such as SDPO, uses the next user message as hindsight evidence. After reading it, the model re-scores the preceding response, and the resulting token-level score change is distilled back into the model. Although distilling this score change can yield strong empirical gains, we show that the change itself contains a systematic confound: even empty templates or unrelated messages induce sizable, response-dependent score shifts. Directly distilling the full score change therefore writes this component into the update as well, which is unrelated to the current feedback. Theoretically, we show that this component is independent of the message and enters the score additively, so it cancels when the scores under the real message are contrasted with those under randomly sampled reference messages, without having to estimate it. Building on this result, we propose CoSD (Contrastive Self-Distillation), which distills how much more the real message shifts the model's scores than a few reference messages sampled from other interactions, requiring no external reward model, preference annotations, or additional generation. Furthermore, CoSD allows cross-session user history to inform how the current message is interpreted. Since message-independent effects of the shared context cancel in the contrast, all branches can condition on the user's history without distilling the user-dependent normalization it introduces. Extensive experiments across multiple backbone models, two real-world interaction corpora, and both on- and off-policy training show that CoSD improves over the initial model and outperforms SDPO.[1]

1 INTRODUCTION

Large language models (LLMs) deployed in production generate a large volume of user interactions. Users rarely provide explicit scores for the responses, but their follow-up messages may correct errors, request different styles, or relay execution outcomes, making these messages a cheap yet informative learning signal that requires no human annotation. Self-distillation provides a direct way to exploit this. SDPO (Hübotter et al., 2026; Buening et al., 2026) compares the LLM's token log-probabilities on the original response before and after reading the follow-up message, and distills the difference back into the policy as an advantage. The entire process relies solely on the model itself, requires no preference pairs or reward model, and can operate directly on deployment logs.

However, the score change after reading a message does not arise entirely from its content. We find even an empty feedback template produces a response-varying score shift, and that filling in unrelated messages from other interactions yields shifts of comparable magnitude to the real follow-up. After the empty-template shift is subtracted, each reply retains a residual shift. This residual is not noise, since two disjoint sets of reference messages reproduce it almost exactly reply by reply, and it varies across replies, so no single constant removes it. Directly distilling the full score therefore writes a message-independent part into the update alongside the actual feedback signal.

[1] Code is available at https://anonymous.4open.science/r/CoSD .

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Based on this observation, we propose CoSD (**Co**ntrastive **S**elf-**D**istillation), which replaces the comparison target. Instead of comparing the real-message score against the original policy, CoSD compares it against scores under several reference messages sampled from other interactions with length matching. Since both branches share the response and the template, any additive term that does not depend on the message content cancels exactly without having to be estimated, leaving only the difference introduced by the real message relative to the reference background. Like SDPO, this method still requires no reward model, preference annotation, or additional generation.

The contrastive construction also enables safely incorporating the user's cross-session history. In real deployments, the same user often has multiple sessions whose records are informative for interpreting the current feedback, but in SDPO the history simply becomes part of the message-independent background that is distilled along with the feedback (Section 4.3). In CoSD the history occupies the same position in all branches, so whatever it contributes to that background cancels, and only its effect on how the current message is read remains. This lets the model judge feedback meaning with richer user context during training, while deployment does not require this history.

We validate this design at three levels. On two real-world log corpora and multiple backbone models, CoSD improves the initial model on the judge-based metrics in nearly all settings, whereas SDPO sometimes degrades or collapses. Learning from logs is off-policy; with a simulated user, we also train on-policy, on responses generated by the model being updated, and CoSD outperforms SDPO in both settings. When every real message is replaced with an unrelated one, CoSD stays at the initial model while SDPO collapses. Messages that carry no information about the response and resemble the reference messages leave CoSD's update near zero. Our contributions are as follows:

- We identify a message-independent and reproducible component in the feedback-driven self-distillation signal, and provide a theoretical account under a user utterance model.
- We propose CoSD, which cancels this by comparing real versus reference message, and show that the contrastive form enables safe incorporation of cross-session user history.
- Across real-world logs and on- and off-policy training with a simulated user, CoSD yields clear improvements over SDPO and in some settings outperforms methods that use additional generation or an external reward model (Appendix K).

2 Related Work

2.1 Self-Distillation for Large Language Models

Context-based self-distillation uses the model's own enhanced performance under additional information as a training signal. Context distillation (Snell et al., 2022) and on-policy distillation (Agarwal et al., 2024) establish the foundations. Recent work lets the model serve as both teacher and student under different contexts, splitting into two lines by the type of privileged information. The first conditions the teacher on correct answers or demonstrations (Zhao et al., 2026; Shenfeld et al., 2026). The second uses feedback obtained after the response is produced. SDPO distills feedback-conditioned scores back into the policy (Hübotter et al., 2026), and subsequent work extends this to natural user follow-up messages in deployment logs (Buening et al., 2026). We keep this scenario but change what the feedback-conditioned score is compared against.

2.2 Learning from User Feedback

The classical path from human feedback to policy improvement trains a reward model on human comparisons and optimizes via reinforcement learning (Ouyang et al., 2022). More recent methods leverage the follow-up messages users produce naturally. One approach trains reward models from in-the-wild feedback (Peng et al., 2026). Another constructs preference data by extracting feedback polarity (Don-Yehiya et al., 2025), inducing user preferences from conversations (Shi et al., 2026), or combining feedback-guided rewriting with user-conditioned reward modeling (Jin et al., 2025). A third preserves the linguistic content of feedback, using it to condition generation (Liu et al., 2024) or to filter rewrites for imitation learning (Scheurer et al., 2024), though feedback reliability as a training signal varies across tasks (Liu et al., 2025). SDPO applied to user interactions takes a different route, directly using raw follow-up messages to construct per-token self-distillation signals on the original response without additional generation or explicit label extraction.

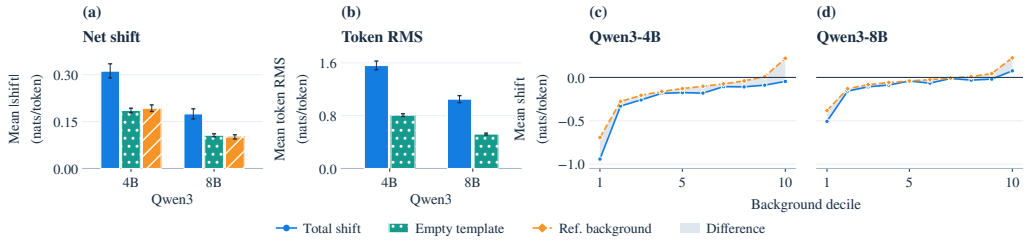

Figure 1: Score shifts relative to empty template and reference background (2,000 WildChat replies per model, 25 references each). (a) Mean absolute reply-level shift. (b) Mean per-reply token RMS. (c,d) Signed means by decile of reference background; shading marks total shift minus background.

3 Preliminary Study

In this section, we first define the learning scenario and the SDPO, and then present measurements showing that the resulting training signal contains a sizeable message-independent component.

3.1 Setting and SDPO

Each record in a post-deployment dialogue log is an interaction (u, x, y, o), where user u receives response $y = (y_1, \ldots, y_L)$ in dialogue context x and sends the next message o. We write o^+ for the message paired with y, and $\mathcal{D}$ for the training set. The response y may come from a different model, making the setting off-policy. The same user often appears across sessions. We write H for user u's messages from other sessions and define the user context as $c := (H, x)$.

SDPO uses a fixed prompt template $T(\cdot)$ to let the model re-score y after reading the follow-up. The token-level advantage is the score difference before and after reading the message:

$$V_t(o) := s_t(o) - s_t^{\varnothing}, \tag{1}$$

where $s_t(o) := \log \pi_{\bar{\theta}}(y_t \mid c, T(o), y_{<t})$ and $s_t^{\varnothing} := \log \pi_{\bar{\theta}}(y_t \mid c, y_{<t})$. The teacher $\pi_{\bar{\theta}}$ shares parameters with π_θ but receives no gradients. $V_t(o^+)$ serves as the training coefficient on $\log \pi_\theta(y_t \mid c, y_{<t})$, pushing tokens with positive advantage up and those with negative advantage down. The training loss is a weighted log-likelihood $\mathcal{L}(\theta) = -\mathbb{E}_{\mathcal{D}}[(1/L)\sum_t \mathrm{sg}[V_t(o^+)] \log \pi_\theta(y_t \mid c, y_{<t})]$, where $\mathrm{sg}[\cdot]$ denotes stop-gradient.

The two conditions in Eq. 1 differ simultaneously in whether the model has entered the feedback-reading context and in what the message says. Since the baseline is the score without any message, the *entire* change enters the training signal, including any component independent of message content. We measure this component with two probes. Fixing (c, y) and the template, we score y under the empty template $T(\varnothing)$ to isolate the template's own effect, and under unrelated messages o^- drawn from other interactions to reveal message-independent shifts beyond the template.

3.2 Measurements

Two models (Qwen3-4B and Qwen3-8B (Yang et al., 2025)) score the same 2,000 logged replies sampled from WildChat (Zhao et al., 2024) under three conditions: the real follow-up, the empty template, and the mean over 25 length-matched reference messages, each relative to the no-template baseline. We report the reply-level mean advantage $(1/L)\sum_t V_t(o)$ (average over tokens).

The background is large and varies with the response. Figure 1(a) shows that the reference background and the empty template each produce a mean absolute shift that is a substantial fraction of the real-message shift. Much of the score change SDPO distills can therefore be reproduced without seeing the actual feedback. Since positive and negative token shifts can cancel within a reply, Figure 1(b) also reports the per-reply token RMS (root mean square); the empty template alone reaches roughly half the token-level magnitude of the real message. Figure 1(c,d) sorts replies into deciles by their signed reference background. The total shift tracks the background across

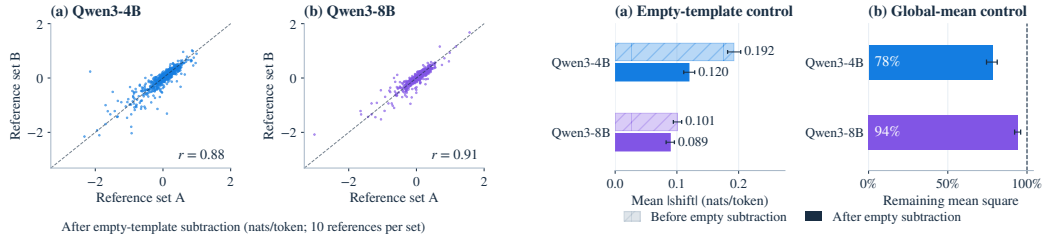

Figure 2: **Left**: repeatability of reference shifts after empty-template subtraction (two disjoint blocks of 10 references). **Right**: (a) mean absolute reference shift before/after empty-template subtraction; (b) background mean square remaining after global-mean centering.

deciles, and the shaded gap between them is the message-specific part. The message-independent component is thus not a fixed offset but varies systematically with the response.

The residual is reproducible and not a constant. Subtracting the empty-template shift from the background leaves a residual whose mean absolute magnitude is of the same order as the background itself (Figure 2, right, panel a). This residual is reproducible: two disjoint blocks of ten reference messages yield per-reply Pearson correlations of 0.88 and 0.91 (Figure 2, left). In contrast, subtracting a single global mean from the background removes only 22% (4B) and 6% (8B) of its mean square (Figure 2, right, panel b). Most of the background therefore varies across replies, motivating a per-reply, message-level contrast (Section 4). Consistently, replacing SDPO's no-message baseline with the empty-template score in training does not help (Table 3).

4 CoSD: Contrastive Self-Distillation

In this section, we explain the origin of the message-independent component found in Section 3.2 (Section 4.1), cancel it with CoSD (Section 4.2), add cross-session user history (Section 4.3), and discuss the design (Section 4.4).

4.1 Decomposition of Feedback-Conditioned Scores

User utterance model. Section 3.2 established the component empirically, which we formalize here. Under an ideal Bayesian teacher, the message-independent part of the SDPO signal is the log-normalizer of the user utterance model, $-\log Z(c, y)$. Following Hübotter et al. (2026), we write the probability that a user says message o after seeing response y in the Boltzmann form:

$$p(o \mid c, y) = \exp r(o; c, y) \,/\, Z(c, y), \qquad Z(c, y) := \textstyle\sum_{o'} \exp r(o'; c, y), \tag{2}$$

where $r(o; c, y)$ is the unnormalized compatibility score between o and y, and $Z(c, y)$ is the normalizing constant. Taking logarithms, $\log p(o \mid c, y) = r(o; c, y) - \log Z(c, y)$. The split between r and $\log Z$ has a free additive constant in (c, y) that does not affect the distribution. All conclusions below use only that $\log Z(c, y)$ does not depend on o and are invariant to this choice (Appendix A).

Ideal Bayesian teacher. Suppose the teacher treats the message in the template as an observation generated after y and rescores accordingly:

$$\log \pi(y \mid c, T(o)) = \log \pi(y \mid c) + \log p(o \mid c, y) - \log p(o \mid c), \tag{3}$$

where $p(o \mid c) := \sum_{y'} \pi(y' \mid c)\, p(o \mid c, y')$.

Three terms of the SDPO signal. The decomposition is stated at sequence level. Writing $V(o) := \sum_t V_t(o)$ and $\kappa(c, o) := -\log p(o \mid c)$, substituting Eq. 3 and expanding $\log p$ via Eq. 2 gives:

$$V(o^+) = \underbrace{r(o^+; c, y)}_{\text{(i) depends on } o^+ \text{ and } y} \quad \underbrace{-\log Z(c, y)}_{\text{(ii) depends only on } y} \quad + \quad \underbrace{\kappa(c, o^+)}_{\text{(iii) constant in } y} . \tag{4}$$

Term (i) captures how well the user's message fits this response. Term (iii) depends only on o^+ and is constant with respect to y. Term (ii) depends only on y, not the message, yet it varies across responses and is therefore optimized as if it were a reward, regardless of what the user actually said.

Reference background and template term. Applying the same decomposition to a reference message $o^- \sim Q$ gives:

$$V(o^-) = r(o^-; c, y) - \log Z(c, y) + \kappa(c, o^-). \tag{5}$$

Taking the expectation over Q leaves $-\log Z(c, y)$ untouched, since it does not depend on the message. The expected reference advantage $\mathbb{E}_Q[V(o^-)]$ is the quantity measured in the preliminary study. Any further additive term that does not depend on the message (e.g., from the actual teacher deviating from Eq. 3) appears identically in $V(o^+)$ and $V(o^-)$ and is handled the same way.

4.2 The Contrastive Self-Distillation Objective

CoSD shares the same loss as SDPO but replaces the advantage. We define the reference distribution $Q(\cdot \mid c, \ell)$ as the distribution of follow-up messages from other interactions in the training corpus, filtered to match the length $\ell = |o^+|$ of the real message. The writers of these messages have not seen the current response y. Sampling $o_1^-, \ldots, o_k^- \overset{\text{iid}}{\sim} Q(\cdot \mid c, |o^+|)$, the CoSD advantage is:

$$\widehat{C}_t := s_t(o^+) - (1/k) \textstyle\sum_{j=1}^{k} s_t(o_j^-). \tag{6}$$

The training loss is:

$$\mathcal{L}(\theta) = -\,\mathbb{E}_{(c,y,o^+)\sim\mathcal{D}}\big[(1/L) \textstyle\sum_{t=1}^{L} \mathrm{sg}[\widehat{C}_t] \, \log \pi_\theta(y_t \mid c, y_{<t})\big], \tag{7}$$

where L is the number of tokens in y and $\mathrm{sg}[\cdot]$ denotes stop-gradient. The branch in Eq. 7 that receives gradients is the *update branch*. The *real branch*, which reads $T(o^+)$, and the k *reference branches*, which read $T(o_j^-)$, compute $\widehat{C}_t$ without gradients; we refer to them collectively as the *scoring branches*.

Cancellation identity. Adding and subtracting $s_t^{\varnothing}$ in Eq. 6 gives $\widehat{C}_t = V_t(o^+) - (1/k) \sum_j V_t(o_j^-)$, so the CoSD advantage is the SDPO advantage minus the mean SDPO advantage of the reference messages. More generally, if $s_t(o) = a_t(c, y) + e_t(o; c, y)$ where a_t does not vary with o, then

$$\widehat{C}_t = e_t(o^+; c, y) - (1/k) \textstyle\sum_{j=1}^{k} e_t(o_j^-; c, y). \tag{8}$$

Any message-independent additive term vanishes exactly for any $k \geq 1$, without estimating a_t and without any model assumptions. This holds token by token.

What is cancelled under the utterance model. Subtracting the reference decomposition (Eq. 5) from Eq. 4 for each reference and averaging gives the sequence-level advantage

$$\widehat{C} = r(o^+; c, y) - (1/k) \textstyle\sum_{j=1}^{k} r(o_j^-; c, y) + \underbrace{\kappa(c, o^+) - (1/k) \textstyle\sum_{j=1}^{k} \kappa(c, o_j^-)}_{\Delta\kappa, \text{ does not depend on } y}. \tag{9}$$

The term $-\log Z(c, y)$ cancels exactly for every k, as does any further message-independent term such as a template-induced perturbation. Taking the expectation over Q replaces the averages by $\mathbb{E}_Q[\cdot]$; the estimator is unbiased with variance $(1/k)\,\mathrm{Var}_{o^-\sim Q}[r(o^-; c, y) + \kappa(c, o^-)]$.

The training coefficient on y is therefore the compatibility of the real message relative to the reference messages, shifted by the y-independent constant $\Delta\kappa$. The choice of Q sets the reference level $(1/k) \sum_j r(o_j^-; c, y)$ against which the real message is compared, whereas the cancellation of $-\log Z(c, y)$ holds for any Q. We match reference lengths to $|o^+|$ so that this reference level is not dominated by message length. Figure 3 shows the effect on a logged example. The user asks to remove the conclusion; SDPO assigns positive advantage to the token "Conclusion" because the background shift is positive at that position, whereas CoSD assigns it negative advantage.

4.3 Shared User Context and Three Branches

In CoSD, all branches share the same user context $c = (H, x)$ and differ only in the message read.

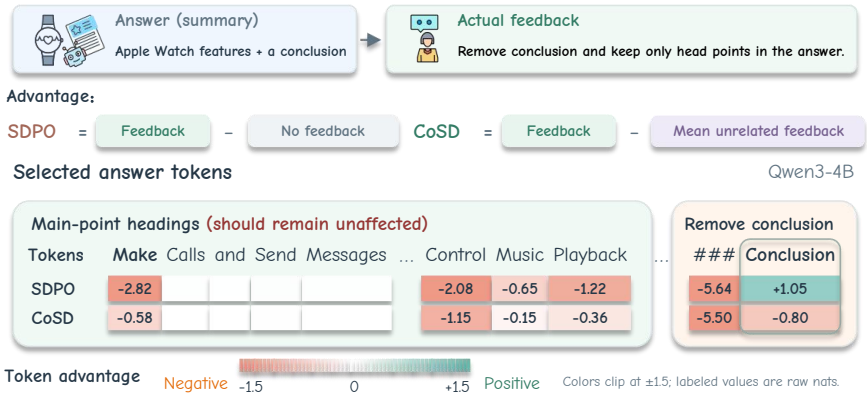

Figure 3: **Token-level advantages on a real example.** SDPO pushes up "Conclusion" because of the message-independent background; CoSD cancels it through the reference contrast. For feedback-irrelevant tokens (i.e., main-point headings), SDPO assigns more negative advantages.

History construction. H consists of user u's messages from other sessions (no assistant replies), placed before the current dialogue x at the same position in all branches (template details in Appendix C).

History enters only through message interpretation. H is a block of user text that is not paired with the current response y. Once H enters the conditioning context, it changes scores in two ways that are unrelated to o^+. (1) The model's overall score on y shifts by $g(c, y) := \log \pi(y \mid H, x) - \log \pi(y \mid x)$. (2) The normalizing constant of the utterance model changes from $-\log Z(x, y)$ to $-\log Z(H, x, y)$, making term (ii) in Eq. 4 user-dependent. Both vary with y and contain no information about o^+. If H were added to SDPO, it would enter c on both sides of Eq. 1, so (1) would cancel while (2) would remain and be optimized as a user-dependent reward.

In CoSD, the scoring branches differ only at $T(o)$ while H occupies the same position in all of them. Any influence of H that does not vary with the message, whether (1), (2), or any deviation of the actual model from Eq. 3, takes the form $a_t(H, x, y)$ in Eq. 8 and cancels token by token. Under the utterance model, Eq. 9 holds with $c = (H, x)$, so the advantage becomes $r(o^+; H, x, y) - (1/k) \sum_j r(o_j^-; H, x, y) + \Delta\kappa$. H changes the model's judgment of how well each message fits this response. For instance, if the user repeatedly requested bullet-point formatting in other sessions and the current o^+ says "too long," the model with H more likely interprets this as a formatting rather than content complaint, shifting token-level credit to different positions in y.

Update branch. The update branch also conditions on $c = (H, x)$, so that the advantage and the updated distribution are defined on the same condition. This is consistent with the discipline that all branches differ only in the message. At deployment, $H = \varnothing$. Since parameters are shared across all users, the learned changes reside in shared parameters rather than user-specific ones. We further examine the performance gap when the update branch omits H (Appendix F.3).

4.4 Discussion

Comparison with constant centering. Batch-mean centering subtracts a single constant per step and therefore removes only the part of $-\log Z(c, y)$ that is shared across responses. Section 3.2 shows that global-mean subtraction leaves 78% (4B) and 94% (8B) of the background mean square intact, since most of it varies across replies. Eq. 9 cancels $-\log Z(c, y)$ per sample, without estimating any centering statistic.

Follow-ups that are not feedback. Some follow-ups carry no feedback about y, for example when the user starts an unrelated task. If such a class of messages is independent of y and distributed like the reference population, its expected contribution to the CoSD update is zero, so CoSD handles it without filtering or classifying message types (Appendix A.1).

Table 1: Main results on real-world logs (WC = WildChat, WF = WildFeedback). Log-only methods (no additional generation). Bold = best within each backbone×corpus block and strictly above Base.

Backbone	Train data	Method	AlpacaEval LC	Arena-Hard	Arena Creative	IFEval	MMLU-Pro
Qwen3-4B	–	Base	32.75	6.50	9.05	78.19	58.47
	WC	SFT	15.75	3.61	4.81	76.34	52.85
		SDPO	42.88	6.07	11.63	74.31	60.53
		CoSD	**45.41**	**7.17**	**15.76**	76.52	**61.59**
	WF	SFT	9.62	2.52	2.57	76.34	50.28
		SDPO	33.85	5.52	9.53	**80.04**	58.24
		CoSD	**44.56**	**6.73**	**11.40**	79.48	**60.76**
Qwen3-8B	–	Base	44.47	9.02	17.98	81.52	63.00
	WC	SFT	21.21	5.44	7.29	80.59	59.67
		SDPO	42.08	7.56	8.96	80.04	61.85
		CoSD	**57.22**	**9.20**	**23.39**	**82.62**	**64.31**
	WF	SFT	19.69	4.17	5.86	77.82	58.07
		SDPO	35.13	7.71	11.82	81.15	61.34
		CoSD	**52.18**	**9.75**	**21.90**	**81.89**	**63.88**
Olmo-3-7B-Instruct	–	Base	43.70	5.39	15.91	81.15	53.57
	WC	SFT	36.63	**5.63**	13.55	**82.81**	52.44
		SDPO	46.88	5.36	17.76	82.26	**54.32**
		CoSD	**47.67**	5.20	**21.51**	81.70	54.23
	WF	SFT	38.80	3.78	14.10	82.99	45.10
		SDPO	45.37	5.31	18.19	80.78	53.95
		CoSD	**47.33**	**5.87**	**18.22**	**83.55**	**54.17**

Appendix F.4 tests both points: batch-mean centering of the SDPO advantage does not improve SDPO, and CoSD trained with random messages stays near Base. Appendix B gives the pseudocode.

5 Experimental Setup

Real-world log experiments. We train on about 50,000 interaction triples from each of two public dialogue corpora, WildChat (Zhao et al., 2024) and WildFeedback (Shi et al., 2026), following the data scale of Buening et al. (2026). Three backbone models are evaluated: Qwen3-4B, Qwen3-8B, and Olmo-3-7B-Instruct (Olmo et al., 2026). All runs use the same recipe: effective batch size 128, 2 epochs, learning rate 2×10^{-6}, and $k = 3$ length-matched reference messages per sample.

Baselines. Log-only methods that require no additional generation: the initial **Base** model, **SFT** (supervised fine-tuning on the logged responses), and **SDPO**. Methods that additionally generate: self-revision SFT (**SRSFT**), **DPO** (Rafailov et al., 2023), and **SimPO** (Meng et al., 2024) all start from a self-revision step, in which the base model rewrites its logged response conditioned on the user's follow-up message. SRSFT fine-tunes on the revision; DPO and SimPO use it as the preferred response and the logged response as the rejected one. All methods share the recipe above; Appendix H sweeps their main hyperparameters and adds common regularizers to SDPO.

Evaluation. Five benchmarks are used: AlpacaEval length-controlled win rate (LC) (Dubois et al., 2025), Arena-Hard (AH) and Arena creative writing (AC) (Li et al., 2025), IFEval prompt-strict accuracy (IF-s) (Zhou et al., 2023), and MMLU-Pro 5-shot (Wang et al., 2024). LC, AH, and AC are judge-based open-ended win rates, directly measuring the training objective. IF-s and MMLU are held-out capability tests. The judge for win-rate benchmarks is GPT-5-mini (Singh et al., 2026).

On- and off-policy experiments. The logged responses above come from other LLMs, so the log experiments are off-policy. To test CoSD when responses are generated on-policy, we use a simulated user who replies to each new response, and compare on-policy training with an off-policy counterpart under the same setup. The task is TL;DR summarization (Stiennon et al., 2022), with two user personas: "more formatting," who prefers lists and markdown, and "commit to one answer," who wants a single best answer rather than more options. We select two backbones from different families than the judge: Olmo-3-7B-Instruct and Gemma3-12B (Team et al., 2025), with 250 interactions per arm. The user simulator and judge are both Qwen3-8B, following the setup of Buening et al. (2026). In the off-policy setting, responses are pre-generated by DeepSeek-V4-Flash-0731 (DeepSeek-AI et al., 2026) and held fixed through training. Full configuration is in Appendix I.

Table 2: Comparison with generation-requiring baselines on three open-ended win-rate metrics. Bold = best in each backbone×corpus block.

		WildChat			WildFeedback		
Backbone	**Method**	**AlpacaEval LC**	**Arena-Hard**	**Arena Creative**	**AlpacaEval LC**	**Arena-Hard**	**Arena Creative**
Qwen3-4B	SRSFT	17.90	3.43	3.14	14.46	3.14	3.84
	DPO	36.75	6.79	8.12	37.40	6.04	9.91
	SimPO	40.35	**7.17**	12.09	36.80	6.44	10.25
	CoSD	**45.41**	**7.17**	**15.76**	**44.56**	**6.73**	**11.40**
Qwen3-8B	SRSFT	26.54	5.22	8.64	26.01	4.88	7.32
	DPO	52.02	9.01	22.83	47.45	9.17	17.84
	SimPO	53.30	9.13	22.62	49.99	9.52	**22.72**
	CoSD	**57.22**	**9.20**	**23.39**	**52.18**	**9.75**	21.90
Olmo-3-7B-Instruct	SRSFT	39.77	5.34	13.90	38.93	4.66	12.81
	DPO	46.46	4.71	18.28	45.32	5.02	17.44
	SimPO	44.72	**5.87**	20.44	46.15	5.72	17.73
	CoSD	**47.67**	5.20	**21.51**	**47.33**	**5.87**	**18.22**

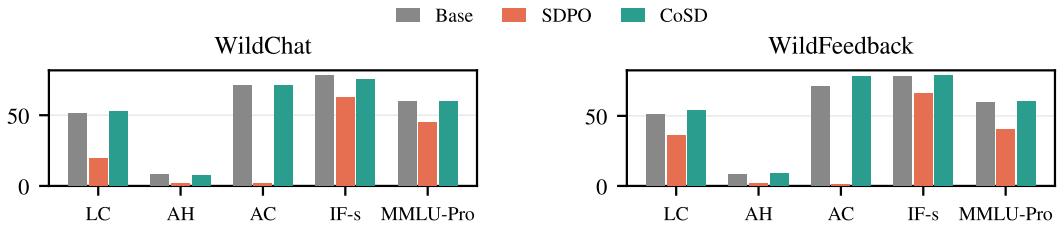

Figure 4: Gemma3-12B results. SDPO degrades on all metrics. Values in Table 14.

6 Experimental Results

6.1 Real-World Log Experiments

Table 1 reports results for three backbones on two corpora. CoSD improves over Base on AlpacaEval LC and Arena Creative in all six settings and on Arena-Hard in five. SDPO improves LC over Base on Qwen3-4B and Olmo-3-7B-Instruct but lowers Arena-Hard in every setting, and on Qwen3-8B it falls below Base on all three judge-based metrics. SFT on the logged responses lowers LC on every backbone and most other metrics, confirming that the logged responses are not suitable as imitation targets. IF-s and MMLU-Pro are largely preserved by CoSD, so the gains on open-ended benchmarks do not come at the expense of instruction following or knowledge.

Comparison with generation-requiring baselines. Table 2 compares CoSD against methods that additionally generate new responses (SRSFT, DPO, SimPO) on the three judge-based win-rate metrics. CoSD leads on AlpacaEval LC in all six backbone×corpus settings and leads on Arena Creative in five of six, despite using only the logged responses. On Qwen3-8B, DPO and SimPO are competitive on some metrics, but CoSD maintains the overall lead.

Robustness on Gemma3-12B. We additionally train Gemma3-12B (Team et al., 2025) on both corpora with the same recipe. Figure 4 shows that SDPO degrades sharply on Gemma3-12B, including on IFEval and MMLU-Pro, which do not depend on a judge, while CoSD improves LC on both corpora and all five metrics on WildFeedback. We investigate the cause in Appendix J.

6.2 On-Policy and Off-Policy Experiments

Figure 5 reports win rates against Base on Olmo-3-7B-Instruct; results on Gemma3-12B are in Appendix I. At the final checkpoint CoSD exceeds both SDPO and Base in all four cells, while SDPO ends below Base in three. On-policy, SDPO improves under "more formatting" (43.93 to 61.96) but declines under "commit to one answer" (50.00 to 33.33), while CoSD improves under both, to 90.35 and 87.50. Off-policy the gap is larger: SDPO stays below Base throughout for both personas (at most 38.10), while CoSD rises to 96.99 and 69.57.

Table 3: Component ablation (Qwen3-4B × WildChat).

Setting	Contrast	References	History	LC	AH	AC
SDPO				42.88	6.07	11.63
SDPO, empty-template baseline				39.86	5.71	7.34
Contrastive only	✓	k=1 random		43.76	6.50	13.35
History only			✓	43.18	6.11	11.51
Contrast + History (k=1)	✓	k=1 random	✓	44.55	6.52	14.49
Contrast + History (k=3)	✓	k=3 random	✓	45.07	7.07	14.27
CoSD (full)	✓	k=3 matched	✓	**45.41**	**7.17**	**15.76**

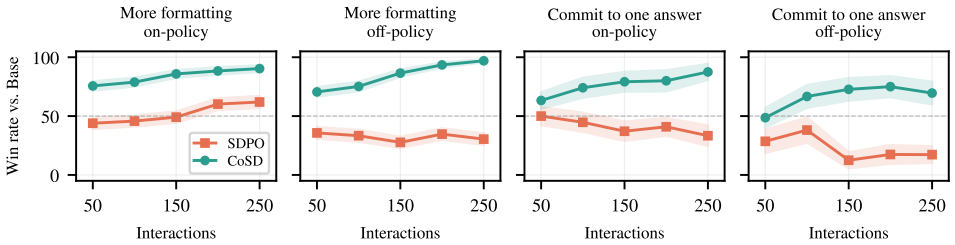

Figure 5: On-policy and off-policy experiments on Olmo-3-7B-Instruct. Win rate vs. Base over 250 interactions. Shading = ±1 SE. Dashed line = 50%.

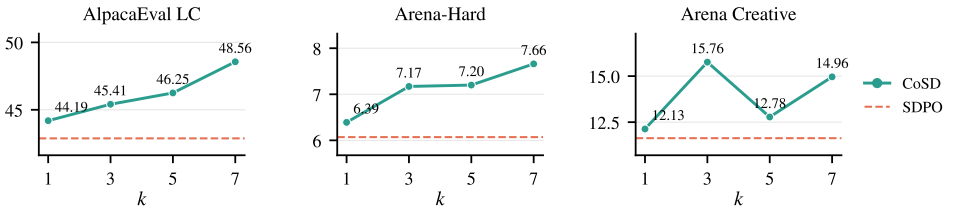

Figure 6: Number of reference messages k in CoSD (Qwen3-4B × WildChat). Dashed line: SDPO.

6.3 Ablation Study

Unless noted otherwise, ablations use Qwen3-4B on WildChat with the main-experiment recipe. The appendix reports further ablations: removing or swapping the user history, filtering semantically related reference messages, conditioning the update branch without history, batch-mean centering of the SDPO advantage, and training with random messages (Appendix F).

Component ablation. Table 3 compares variants that differ in which components are active. Replacing SDPO's no-message baseline with the empty-template score, which removes only the template's own contribution to the shift, lowers all three metrics. The contrastive objective alone (k=1, no history) improves over SDPO on all three metrics. Adding user history to SDPO alone changes LC from 42.88 to 43.18; combined with the contrast it reaches 44.55. Increasing k and adding length matching further raise LC. Figure 6 sweeps k for the full method: LC increases monotonically from k=1 to k=7, at a scoring-side cost linear in k (Appendix G); all main experiments use k=3.

7 Conclusion

In this work, we identified a message-independent, response-varying component in the feedback-conditioned self-distillation signal. CoSD removes it by contrasting real-message scores with reference-message scores, which also lets cross-session user history enter as shared context. Across backbones, real-world corpora, and both on- and off-policy training, CoSD improves over the initial model and SDPO without a reward model or additional generation.

AI USE STATEMENT

In this work, we used generative AI tools to assist in checking the proofs, and to assist in implementing methods (refining code). We have not used generative AI tools for the other required-disclosure tasks: developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for or writing proofs, proposing or refining hypotheses, designing the research methodology or experiments, translation, cleaning or reformatting datasets, or interpreting results; qualitative and thematic data analysis is not applicable to this work. The model-generated responses and simulated user feedback in the on- and off-policy experiments (Section 5, Appendix I) are the subject of the research, as part of the experimental method, rather than AI assistance in conducting it. Additionally, we used generative AI tools to identify relevant literature and to edit the paper to improve readability. We have reviewed all AI-assisted work: every checked derivation was re-derived by the authors, AI-assisted code was reviewed and tested by the authors, every suggested related work was read and verified and cited by the authors, and edited text was checked against the original intent. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we released our implementation of CoSD, including data preprocessing, training, and evaluation scripts, at https://anonymous.4open.science/r/CoSD . The training corpora (WildChat and WildFeedback), the TL;DR task, and the evaluation benchmarks (AlpacaEval, Arena-Hard, IFEval, and MMLU-Pro) are all publicly available, and the benchmarks come with official evaluation code. The training recipe is given in Section 5, the training algorithm and templates in Appendices B and C, and the full configuration of the on- and off-policy experiments in Appendix I. We hope these measures facilitate the replication of our results and support future work on learning from real user interactions.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions, 2026. URL https://arxiv.org/abs/2603.12273.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL https://arxiv.org/abs/2507.06261.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chengyu Hou, Chenhao Xu, Chenze Shao, Chong Ruan, Conner Sun, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Donghao Li, et al. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. Naturally occurring feedback is common, extractable and useful, 2025. URL https://arxiv.org/abs/2407.10944.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators, 2025. URL https://arxiv.org/abs/2404.04475.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

Chuanyang Jin, Jing Xu, Bo Liu, Leitian Tao, Olga Golovneva, Tianmin Shu, Wenting Zhao, Xian Li, and Jason Weston. The era of real-world human interaction: Rl from user conversations, 2025. URL https://arxiv.org/abs/2509.25137.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/li25h.html.

Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=6xfe4IVcOu.

Yuhan Liu, Michael JQ Zhang, and Eunsol Choi. User feedback in human-LLM dialogues: A lens to understand users but noisy as a learning signal. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2666–2681, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.133. URL https://aclanthology.org/2025.emnlp-main.133/.

Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, and Yang Liu. Skywork-reward-v2: Scaling preference data curation via human-ai synergy. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 133805–133838, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d8836d0294651a867c2ca51012923997-Paper-Conference.pdf.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 124198–124235. Curran Associates, Inc., 2024. doi: 10.52202/079017-3946. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e099c1c9699814af0be873a175361713-Paper-Conference.pdf.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, et al. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Hao Peng, Yunjia Qi, Xiaozhi Wang, Zijun Yao, Lei Hou, and Juanzi Li. Wildreward: Learning reward models from in-the-wild human interactions, 2026. URL https://arxiv.org/abs/2602.08829.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. doi: 10.52202/075280-2338. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale, 2024. URL https://arxiv.org/abs/2303.16755.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

Taiwei Shi, Zhuoer Wang, Longqi Yang, Ying-Chun Lin, Zexue He, Mengting Wan, Pei Zhou, Sujay Jauhar, Sihao Chen, Shan Xia, Hongfei Zhang, Jieyu Zhao, Xiaofeng Xu, Xia Song, and Jennifer Neville. Wildfeedback: Aligning llms with in-situ user interactions and feedback, 2026. URL https://arxiv.org/abs/2408.15549.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, et al. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. URL https://arxiv.org/abs/2209.15189.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2022. URL https://arxiv.org/abs/2009.01325.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Changyue Wang, Weihang Su, Qingyao Ai, Xingzhao Yue, Rui Zhang, Xiaojia Chang, and Yiqun Liu. Improve large language model systems with user logs, 2026. URL https://arxiv.org/abs/2602.06470.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models, 2026. URL https://arxiv.org/abs/2601.18734.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Derivation Details

A.1 Non-Feedback Messages Under Mixture Models

The user's next message after seeing y can arise from different behavioral modes: correcting an error, asking for a different style, thanking the assistant, or starting an unrelated task. We model this with a latent variable $z \in \{1, \dots, M\}$ indexing the mode,

$$p(o \mid c, y) = \textstyle\sum_{z=1}^{M} p(z \mid c, y)\, p(o \mid z, c, y), \tag{10}$$

where $p(z \mid c, y)$ is the probability that the user is in mode z after seeing y, and $p(o \mid z, c, y)$ is the message distribution within that mode. The mode z is never observed; only the message o is. The Boltzmann form in Eq. 2 places no restriction on the shape of $p(o \mid c, y)$, so Eq. 10 is a special case of it with $r(o; c, y) = \log p(o \mid c, y) + \log Z(c, y)$ for any choice of the normalizer. In particular, $Z(c, y)$ is defined by the sum over the whole message space and does not depend on how that space is partitioned into modes. The decomposition of $V(o)$ in Eqs. 4 and 5, and hence the cancellation of $-\log Z(c, y)$ in Eq. 9, therefore hold unchanged under the mixture.

Expected update under the mixture. What the mixture structure does affect is the value of the remaining terms. Write $\phi(o; c, y) := r(o; c, y) + \kappa(c, o)$ for the part of $V(o)$ that survives the contrast, so that Eq. 9 reads $\widehat{C} = \phi(o^+; c, y) - (1/k) \sum_j \phi(o_j^-; c, y)$. Taking the expectation over the real message $o^+ \sim p(\cdot \mid c, y)$ and over the reference draws $o_j^- \sim Q(\cdot \mid c, \ell)$, and expanding $p(\cdot \mid c, y)$ via Eq. 10,

$$\mathbb{E}_{o^+, Q}[\widehat{C}] = \textstyle\sum_{z=1}^{M} p(z \mid c, y) \Big(\mathbb{E}_{o \sim p(\cdot \mid z, c, y)}[\phi(o; c, y)] - \mathbb{E}_{o^- \sim Q}[\phi(o^-; c, y)] \Big), \tag{11}$$

where we used $\sum_z p(z \mid c, y) = 1$ to distribute the reference term over modes. The expected update is thus a weighted sum of per-mode brackets, each comparing the mode's messages with the reference population under the same ϕ.

Three cases. Consider a single mode z_0. First, if its messages are unrelated to y and distributed like the reference population, $p(\cdot \mid z_0, c, y) = Q(\cdot \mid c, \ell)$, then the two expectations in its bracket coincide and the bracket is zero. Such a mode contributes nothing to the expected update, regardless of its weight $p(z_0 \mid c, y)$. Starting an unrelated task is the typical example: the message does not depend on y, and messages of this kind are exactly what other interactions' follow-ups contain. Second, if the mode is unrelated to y but its distribution differs from Q, the bracket equals $\mathbb{E}_{z_0}[\phi] - \mathbb{E}_Q[\phi]$ and is in general nonzero. Length matching in Q is designed to reduce this gap for the most common source of mismatch, message length, but it does not remove the gap for other attributes. Third, if the mode does depend on y, as corrections do, its bracket carries the information CoSD is meant to distill. The mixing weights $p(z \mid c, y)$ are retained in Eq. 11.

None of this requires identifying the mode of a given message or estimating $p(z \mid c, y)$; the analysis describes the expected update, not a filtering step. The random-pairing control in Appendix F.4 is the degenerate case of Eq. 10 with a single mode, $p(z_0 \mid c, y) = 1$ and $p(o \mid z_0, c, y) = Q(o \mid c, \ell)$, since o^+ is itself drawn from the reference population. Every bracket in Eq. 11 then vanishes and $\mathbb{E}_{o^+,Q}[\widehat{C}] = 0$ for every (c, y), so the expected update is zero. The trained model staying at Base in that control is consistent with this prediction.

A.2 Zero-Point Freedom and the Interpretation of Term (ii)

Zero-point freedom. Shifting r by a function of (c, y) alone, $r'(o; c, y) = r(o; c, y) + a(c, y)$, leaves the distribution $p(o \mid c, y)$ invariant while translating $\log Z$ by $a(c, y)$. The (i)/(ii) split in Eq. 4 therefore depends on the zero-point convention. CoSD is unaffected because $a(c, y)$ cancels in the real-minus-reference difference. The two properties used in the main text ($\log Z$ does not depend on o, and it is retained by SDPO but cancelled by CoSD) are both convention-free.

What is identified. Because the split is free, neither $-\log Z(c, y)$ nor the average reference compatibility $\mathbb{E}_Q[r(o^-; c, y)]$ is identified on its own; only their sum is. With $r = \log p$, $Z \equiv 1$ and the whole reference average is compatibility; with r shifted so that $\mathbb{E}_Q[r(o^-; c, y)] = 0$, the whole reference average is term (ii). We therefore take the reference background $b_Q(c, y) := \mathbb{E}_{o^- \sim Q}[V(o^-)]$ as the object of the analysis. Its value is convention-free, it is the quantity measured in Section 3.2, and $-\log Z$ is a name for it under one convention. The properties that matter for training are also convention-free: b_Q does not depend on the message the user sent, yet it varies with y, so SDPO adds the same response-dependent offset whatever the feedback says. Under Eq. 3, what CoSD retains is $\log p(o^+ \mid c, y) - \mathbb{E}_Q[\log p(o^- \mid c, y)]$ up to a y-independent constant, i.e., how much better y explains the real message than typical messages from Q, which involves neither r nor Z. Whether distilling b_Q is harmful is an empirical question, answered by the random-message control in Table 10. There every real message is replaced with a message from another interaction, so SDPO's advantage on each sample is a single-draw (k=1) estimate of b_Q, and SDPO distills nothing but the background. LC falls from 32.75 (Base) to 23.87, whereas CoSD, whose expected advantage is then zero, stays at 32.54.

An entropy decomposition of term (ii). For intuition, the normalizer can be written in terms of the user's follow-up distribution. Since $\log p(o \mid c, y) = r(o; c, y) - \log Z(c, y)$ for every o, taking the expectation under $p(\cdot \mid c, y)$ gives

$$\log Z(c, y) = \mathbb{E}_{o \sim p(\cdot \mid c, y)}[r(o; c, y)] + \mathcal{H}\big(p(\cdot \mid c, y)\big), \tag{12}$$

where $\mathcal{H}$ is the Shannon entropy. Term (ii) thus combines the average compatibility of the follow-ups the user might send with the uncertainty over which one they will send. Both are properties of the whole follow-up distribution given y, not of the message o^+ that was actually sent, which is why neither should enter an advantage meant to reflect the specific feedback. The numerical value of each part depends on the zero point of r, since shifting r by $a(c, y)$ shifts $\mathbb{E}_p[r]$ and $\log Z$ by the same amount; the decomposition is therefore a way of reading term (ii), not a basis for comparing its size across responses.

A.3 Sequence-Level and Token-Level Views

The loss in Eq. 7 weights each token by $\widehat{C}_t$, whereas Section 4.1 analyzes the sequence-level sum $\widehat{C} = \sum_t \widehat{C}_t$. This section explains why the decomposition is stated at the sequence level, what the sum controls in training, and what each level of the analysis establishes for CoSD.

Why the decomposition is sequence-level. The utterance model $p(o \mid c, y)$ describes the user's message after reading the complete response, so $r(o; c, y)$ and $Z(c, y)$ are defined only for the full y. Under the ideal-teacher assumption (Eq. 3), the token-level SDPO advantage is the increment in the log-probability of the message as one more token of y is revealed,

$$V_t(o) = \log p(o \mid c, y_{\le t}) - \log p(o \mid c, y_{<t}), \qquad p(o \mid c, y_{\le t}) := \textstyle\sum_{y_{>t}} \pi(y_{>t} \mid c, y_{\le t})\, p(o \mid c, y), \tag{13}$$

where the prefix-conditional message distribution averages the utterance model over the continuations the policy would produce. Summing over t telescopes to $V(o) = \log p(o \mid c, y) - \log p(o \mid c)$, and only at this point does $p(o \mid c, y)$ take the Boltzmann form of Eq. 2, giving the three terms of Eq. 4. The sequence-level sum is therefore the quantity with an interpretation as evidence the complete response provides for the message; the token-level advantages are its increments.

What the sum controls in training. Up to the factor $-1/L$, the gradient of Eq. 7 for one sample is $\sum_t \widehat{C}_t \nabla_\theta \log \pi_\theta(y_t \mid c, y_{<t})$. If all token weights were equal to $\widehat{C}/L$, this would reduce to $\widehat{C}\, \nabla_\theta \log \pi_\theta(y \mid c)/L$, a sequence-level policy gradient with $\widehat{C}$ as the return; in general, the differences among the $\widehat{C}_t$ also shape the update. For SDPO, the message-independent parts $a_t(c, y)$ sum to $-\log Z(c, y)$ up to a constant, which is the total offset in SDPO's advantage regardless of the message, and the profile of the a_t across positions decides where that offset lands. Figure 3 shows one such profile; the background is far from uniform across tokens.

What each level establishes for CoSD. Eq. 8 is a token-level identity: at every position, $\widehat{C}_t$ retains only the difference of the message-dependent parts e_t, and the message-independent part a_t is removed without being computed. Eq. 9 is its sequence-level consequence under the utterance model: the removed parts sum to $-\log Z(c, y)$, so the total offset SDPO would have applied is gone. The token-level identity is what guarantees that the cancellation happens at every position and not merely on average; the sequence-level analysis is what identifies the removed quantity.

B Training Algorithm

Algorithm 1 CoSD Training

Require: Training set $\mathcal{D} = \{(u_i, x_i, y_i, o_i^+)\}$, reference count k, learning rate η

for each mini-batch $\{(u_i, x_i, y_i, o_i^+)\}_{i=1}^B$ **do**
  Build user history H_i and context $c_i = (H_i, x_i)$ for each sample
  Sample $o_{i,1}^-, \ldots, o_{i,k}^- \sim Q(\cdot \mid c_i, |o_i^+|)$ with length matching
  Scoring branches (no grad): compute $s_{i,t}(o_i^+)$ and $s_{i,t}(o_{i,j}^-)$ for $j = 1, \ldots, k$
  Compute $\widehat{C}_{i,t} = s_{i,t}(o_i^+) - (1/k) \sum_{j=1}^k s_{i,t}(o_{i,j}^-)$, stop gradient
  Update branch: compute $\log \pi_\theta(y_{i,t} \mid c_i, y_{i,<t})$
  $\mathcal{L} = -(1/B) \sum_i (1/L_i) \sum_t \widehat{C}_{i,t} \log \pi_\theta(y_{i,t} \mid c_i, y_{i,<t})$
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
end for

C Template Text and Sensitivity

The main experiments use the template format from the official SDPO repository.[2] The follow-up message o is injected as a new assistant-role message after the user turn and before the real assistant response. Three semantically equivalent formats are tested. All three wrappers, i.e., the hindsight-context text with an empty message and without the chat-template markup, are exactly 29 tokens under both the Qwen3-4B and Olmo-3-7B-Instruct tokenizers. The hindsight-context text of each format is shown below, with {o} as the message placeholder; the chat-template markup of the assistant-role message that carries it is omitted.

Template 1 (default from SDPO repository)

```
=== HINDSIGHT CONTEXT ===
[The following is a future user message.  Use this to guide your
answer to the user prompt.]
{o}
```

[2] https://github.com/lasgroup/user_interactions

Template 2

```
Advance notice of what happens next:  written below are the very
words that this person sends back afterwards.  Let them shape how
you address their request.
{o}
```

Template 3

```
<<<PREVIEW OF A LATER TURN>>>
(Below sits what the human types later on; build your reply to
their query around it.)
{o}
```

After rendering with the Qwen3-4B chat template, the scoring context has the following layout, where `[hindsight context]` stands for the assistant-role message carrying one of the three texts above:

Rendered scoring context (Qwen3-4B)

```
<|im_start|>user
{x}<|im_end|>
[hindsight context]
<|im_start|>assistant
```

User history injection. The user history H is injected as a system message at the front of the prompt, before the dialogue turns. It uses the user's follow-ups from at most five other conversations, truncated to 100 tokens:

User history context

```
<|im_start|>system
=== USER PROFILE ===
[Things this user has asked for in earlier, unrelated
conversations.  Use this to anticipate what they will want.]
{H}<|im_end|>
...  followed by the dialogue turns of x ...
```

SRSFT revision prompt. For the SRSFT, DPO, and SimPO baselines, the model generates a revised response conditioned on the user's follow-up message using the following prompt:

Self-revision prompt (SRSFT / DPO / SimPO)

```
You will see a conversation between a user and an AI assistant, the
assistant's response, and the user's follow-up message reacting to
that response.

<conversation>
{conversation}
</conversation>

<assistant_response>
{response}
</assistant_response>

<user_followup>
{followup}
</user_followup>

Write an improved version of the assistant's response that takes
the user's follow-up into account.  Output ONLY the improved
```

```
response text.  Do not include explanations, labels, or any text
before or after the response.
```

Table 4 shows that SDPO is catastrophically sensitive to template wording on Qwen3-4B (LC drops from 42.88 to 12.03 and 6.61), while CoSD remains stable across all three formats and both backbones. Table 5 examines the outputs themselves: under the two alternative templates, most SDPO outputs contain long repeated spans and more than half run to the generation cap, while CoSD's repetition rate, cap rate, and length distribution stay at the Template 1 level.

Table 4: Template sensitivity. Three semantically equivalent injection formats.

		SDPO					CoSD				
Backbone	Format	LC	AH	AC	IF-s	MMLU	LC	AH	AC	IF-s	MMLU
Qwen3-4B	Template 1	42.88	6.07	11.63	74.31	60.53	45.41	7.17	15.76	76.52	61.59
	Template 2	12.03	3.13	1.59	31.79	15.58	44.17	5.81	14.14	75.60	60.74
	Template 3	6.61	2.62	1.01	22.74	10.74	44.84	6.80	14.01	74.49	60.73
Olmo-3-7B-Instruct	Template 1	46.88	5.36	17.76	82.26	54.32	47.67	5.20	21.51	81.70	54.23
	Template 2	45.79	4.95	15.13	82.81	54.17	47.00	5.54	19.55	83.36	54.03
	Template 3	46.04	5.02	17.62	83.18	54.40	46.73	5.13	19.62	83.36	53.75

Table 5: Output-side statistics on the 805 AlpacaEval prompts (Qwen3-4B $\times$ WildChat). Repetition: fraction of outputs containing a 5–40 character span repeated ten or more times consecutively. At cap: fraction of outputs of at least 14,400 characters, about 90% of the 4,096-token generation limit. Unterminated: fraction of outputs that do not end with sentence-final punctuation or a closing bracket. Mean chars, p50, and p90 describe the output length distribution in characters.

Method	Format	Repetition	At cap	Unterminated	Mean chars	p50 / p90
SDPO	Template 1	2.73%	0.12%	26.71%	3160	3000 / 5443
SDPO	Template 2	79.01%	59.63%	52.80%	18313	22144 / 30971
SDPO	Template 3	89.32%	80.87%	51.18%	28411	32541 / 40951
CoSD	Template 1	3.23%	0.50%	31.55%	3477	3351 / 6003
CoSD	Template 2	4.60%	0.75%	30.68%	3348	3122 / 5999
CoSD	Template 3	2.48%	0.50%	26.09%	3440	3293 / 5923

D Seed Replication

Table 6 reports results over three seeds (42, 43, 44) for Qwen3-4B on WildChat. Each cell shows the mean $\pm$ standard deviation across the three runs. Δ is the per-seed difference CoSD $-$ SDPO, averaged over seeds, with a 95% confidence interval from the three paired differences. "Sig." indicates whether the 95% CI excludes zero. CoSD improves over SDPO significantly on LC, MMLU-Pro, and AH. AC and IF-s do not reach significance with three seeds.

Table 6: Seed replication (Qwen3-4B $\times$ WildChat, seeds 42/43/44).

Metric	Base	SFT	SDPO	CoSD	Δ [95% CI]	Sig.
AlpacaEval LC	32.87$\pm$0.31	15.63$\pm$1.04	42.73$\pm$0.79	45.34$\pm$0.77	+2.61 [+0.84, +4.38]	Yes
MMLU-Pro	58.32$\pm$0.16	53.07$\pm$0.37	60.53$\pm$0.01	61.35$\pm$0.20	+0.82 [+0.32, +1.32]	Yes
Arena Creative	9.24$\pm$0.42	4.68$\pm$1.43	12.35$\pm$0.77	14.69$\pm$1.47	+2.34 [$-$0.71, +5.39]	No
Arena-Hard	6.43$\pm$0.26	3.76$\pm$0.58	6.32$\pm$0.41	7.36$\pm$0.46	+1.04 [+0.05, +2.03]	Yes
IFEval	77.57$\pm$0.40	76.09$\pm$0.86	75.29$\pm$0.71	76.71$\pm$0.68	+1.43 [$-$0.16, +3.00]	No

E Judge Independence

To verify that the main results do not depend on the choice of judge, we re-evaluate the same four model outputs (Base, SFT, SDPO, CoSD on Qwen3-4B $\times$ WildChat) with three judges: GPT-5-mini

(the main-text judge), Gemini-2.5-Flash (Comanici et al., 2025), and DeepSeek-V4-Flash-0731. All prompts, position swaps, and answer parsers are identical across judges. Table 7 reports AlpacaEval LC, Arena-Hard, and Arena Creative separately. The GPT-5-mini row uses the values from Table 1. On AlpacaEval LC the ranking CoSD $>$ SDPO $>$ Base $>$ SFT is identical under all three judges (pairwise Kendall $\tau_b = +1.00$). On Arena-Hard the first two judges agree on CoSD $>$ Base $>$ SDPO $>$ SFT, while DeepSeek ranks Base first and CoSD second. On Arena Creative, Gemini ranks SDPO above CoSD while the other two rank CoSD first. SFT is consistently last under all judges on all three metrics.

Table 7: Judge independence (Qwen3-4B $\times$ WildChat). Same outputs, different judges.

	AlpacaEval LC				Arena-Hard				Arena Creative			
Judge	Base	SFT	SDPO	CoSD	Base	SFT	SDPO	CoSD	Base	SFT	SDPO	CoSD
GPT-5-mini	32.75	15.75	42.88	**45.41**	6.50	3.61	6.07	**7.17**	9.05	4.81	11.63	**15.76**
Gemini-2.5-Flash	37.33	16.41	41.85	**46.49**	10.42	5.22	10.29	**13.16**	5.08	2.72	**8.11**	7.31
DeepSeek-V4-Flash	33.81	16.84	37.60	**38.58**	**5.70**	3.57	4.48	5.49	4.62	2.71	5.53	**6.19**

F Further Ablation Studies

All experiments in this appendix use the main-experiment recipe.

F.1 User History

Table 8 compares the full method against two history variants across four backbone$\times$corpus settings. Removing history entirely consistently lowers LC. Replacing the user's own history with another user's history also reduces performance in most settings, suggesting that part of the benefit comes from the specific user's history.

Table 8: History ablation (LC / AH). Comparing CoSD against removing history and using another user's history.

	No history		Wrong user's history		CoSD	
Setting	LC	AH	LC	AH	LC	AH
Qwen3-4B $\times$ WC	45.07	6.89	44.73	7.16	**45.41**	**7.17**
Qwen3-4B $\times$ WF	39.34	**6.77**	41.71	6.62	**44.56**	6.73
Olmo-3-7B-Instruct $\times$ WC	46.41	5.11	46.58	**5.85**	**47.67**	5.20
Olmo-3-7B-Instruct $\times$ WF	46.66	5.16	46.69	4.66	**47.33**	**5.87**

F.2 Reference Message Relevance

We measure whether reference messages o^- are "coincidentally relevant" to o^+ using Qwen3-Embedding-0.6B sentence embeddings (Zhang et al., 2025). Among the 150,009 (o^+, o^-) pairs used in training, the median cosine similarity is 0.304, close to the median of 0.287 obtained after randomly permuting the o^+ column. Pairs exceeding the 99th percentile of this permutation control account for 3.71% of the total. To test whether this tail harms training, we filter candidates with cosine ≥ 0.4945 (the 90th percentile of the actual distribution), rejecting 165,305 candidates while maintaining full k=3 coverage. Head-to-head evaluation of the filtered model against the unfiltered model yields win rates of 50.56% (AlpacaEval, 95% CI [47.89, 53.23]) and 49.80% (Arena, [46.86, 52.67]). Both intervals include 50%, indicating no measurable effect of semantic filtering. Filtering also degrades length matching, since rejected candidates are replaced by later entries of the length-sorted pool (pairs with a length ratio above 2$\times$ rise from 0.99% to 3.75%), so the comparison is, if anything, tilted against the filtered model.

F.3 Gradient Branch Alignment

Section 4.3 conditions the update branch on $c = (H, x)$, the same context as the scoring branches, so that the advantage and the distribution being updated are defined on the same condition. The

cost of this choice is a mismatch with deployment, where $H = \varnothing$. We test the alternative: keep the advantage $\widehat{C}_t$ exactly as computed by the scoring branches with H, but apply it to $\log \pi_\theta(y_t \mid x, y_{<t})$, the inference-time condition without H. Only the condition inside the loss changes; data, reference sampling, and all hyperparameters are identical.

Table 9 reports both variants on two backbone $\times$ corpus settings. The two are within a point of each other on Qwen3-4B, and conditioning on (H, x) is ahead by 1.4 LC on Olmo-3-7B-Instruct. The choice therefore does not drive the gains in Table 1; we keep $c = (H, x)$ for the consistency argument in Section 4.3.

Table 9: Gradient branch alignment. Update branch with $c = (H, x)$ vs. $c = x$.

	Update with $c = x$		CoSD ($c = (H, x)$)	
Setting	LC	AH	LC	AH
Qwen3-4B $\times$ WC	45.58	7.09	45.41	**7.17**
Olmo-3-7B-Instruct $\times$ WC	46.31	5.14	**47.67**	**5.20**

F.4 Centering and Random-Pairing Controls

Table 10 reports three controls. Batch-mean centering of the SDPO advantage lowers LC from 42.88 to 41.81, consistent with Section 3.2: a single constant removes only a small part of the background, and centering the remainder introduces noise. SDPO trained with random messages collapses (LC 23.87), since the full score change under an irrelevant message is distilled without correction. CoSD trained with the same random messages returns to the neighborhood of Base (LC 32.54), confirming that the contrastive form does not manufacture a learning direction from the reference distribution.

Table 10: Controls (Qwen3-4B $\times$ WildChat). Centering and random-pairing diagnostics.

Method	Message	AlpacaEval LC	Arena-Hard	Arena Creative
Base	—	32.75	6.50	9.05
SDPO	Real	42.88	6.07	11.63
SDPO + batch-mean centering	Real	41.81	6.19	10.16
SDPO	Empty	3.65	1.38	0.89
SDPO	Random	23.87	4.71	3.77
CoSD	Random	32.54	6.10	8.47

G Computational Overhead

Cost accounting. We count $2N$ FLOPs per token for a forward pass and $4N$ for a backward pass, with one forward recomputed under gradient checkpointing, so the update branch costs $8N$ per token and each scoring branch $2N$. The scoring forwards are executed by the model being trained, so there is no separate teacher and the multiplier is the same for all model sizes. On Qwen3-4B $\times$ WildChat (50,003 rows, 2 epochs), CoSD requires 42.9 TFLOPs per row versus 34.2 for SDPO, a $\times 1.25$ multiplier. Wall-clock time increases from 5.19h to 6.34h ($\times 1.22$), less than the FLOP ratio, since the $k{+}1$ scoring contexts share the prefix c in the KV cache and are packed into one batch. The extra cost comes entirely from the scoring branches (no gradients); the forward-backward pass on the update branch is identical to SDPO.

Hardware and software. All training runs use NVIDIA H20 GPUs. Training is implemented on top of the Hugging Face `transformers` Trainer[3] with bf16 precision, FlashAttention-2,[4] gradient checkpointing, and 8-bit AdamW from `bitsandbytes`.[5] Evaluation generation uses vLLM[6] v0.10.0.

[3] `https://github.com/huggingface/transformers`
[4] `https://github.com/Dao-AILab/flash-attention`
[5] `https://github.com/bitsandbytes-foundation/bitsandbytes`
[6] `https://github.com/vllm-project/vllm`

H Baseline Hyperparameter Sweeps and SDPO Regularization

Provenance of the main-experiment settings. The shared recipe of Section 5 (learning rate 2×10^{-6}, 2 epochs, effective batch 128, cosine schedule with 5% warmup, 8-bit AdamW, bf16, gradient-norm clipping at 10, seed 42) is the default configuration of the official SDPO implementation, taken unchanged so that our SDPO runs are a reproduction rather than a variant; CoSD keeps it as well. DPO uses $\beta = 0.1$, the standard value from Rafailov et al. (2023), with a frozen copy of Base as the reference model, and inherits the optimizer, learning rate, schedule, and batch geometry of the SDPO recipe. SimPO uses $\beta = 2.0$ and $\gamma/\beta = 0.5$, the values recommended by Meng et al. (2024) for instruction-tuned backbones, and imports every component other than the loss from the DPO baseline, so that SimPO and DPO differ in the loss alone. We keep the learning rate at 2×10^{-6} for SimPO rather than adopting the lower value used in its paper, so that the DPO-versus-SimPO comparison does not also change the learning rate; the sweep below includes lower learning rates for both.

Sweep design. Since all baselines share one recipe, a natural concern is that this recipe under-tunes them. We sweep the main hyperparameters of each baseline on Qwen3-4B × WildChat, keeping every other setting of the main recipe fixed. The main-experiment configurations are highlighted in bold, and all cells are evaluated with the same judge and protocol as Table 1. For reference, CoSD reaches 45.41 LC, 7.17 AH, and 15.76 AC in this setting.

SDPO learning rate and training length. Table 11 varies the learning rate over $\{1, 2, 4\} \times 10^{-6}$ and training over 1, 2, and 4 epochs. The main-experiment setting is the best cell of the grid. At 10^{-6} SDPO improves with more epochs but stays below 37 LC. At 4×10^{-6}, and at 2×10^{-6} with 4 epochs, LC falls close to zero. SDPO therefore has a narrow usable region, and its main-experiment setting sits at its upper edge rather than in an under-tuned corner.

Table 11: SDPO learning rate × training length (Qwen3-4B × WildChat). Bold: main-experiment setting.

	LC			AH			AC		
LR \ epochs	1	2	4	1	2	4	1	2	4
1×10^{-6}	32.82	35.80	36.95	6.23	5.87	6.76	9.35	8.90	10.25
2×10^{-6}	38.91	**42.88**	0.37	6.16	**6.07**	0.98	9.85	**11.63**	0.81
4×10^{-6}	0.05	0.00	2.49	0.46	0.14	1.11	1.32	0.47	0.30

DPO and SimPO learning rate and β. Table 12 varies the learning rate over $\{0.5, 1, 2\} \times 10^{-6}$ and β over three values each, with $\gamma/\beta = 0.5$ fixed for SimPO. DPO is flat across its grid (33.6 to 36.8 LC), and its main-experiment setting is the best cell. SimPO improves with larger β at the highest learning rate and peaks at 42.02 LC, 1.7 points above its main-experiment setting and still below SDPO (42.88) and CoSD (45.41). Tuning therefore does not change the ordering in Table 2.

Table 12: DPO and SimPO learning rate × β (Qwen3-4B × WildChat). Bold: main-experiment setting. SimPO uses $\gamma/\beta = 0.5$.

	LC			AH			AC		
DPO, LR \ β	0.05	0.1	0.3	0.05	0.1	0.3	0.05	0.1	0.3
5×10^{-7}	35.86	35.33	33.64	5.50	6.24	5.89	10.15	9.16	11.00
1×10^{-6}	35.80	35.65	35.93	6.19	6.34	5.90	8.87	9.03	8.95
2×10^{-6}	36.54	**36.75**	35.04	6.68	**6.79**	6.53	11.62	**8.12**	8.53
SimPO, LR \ β	2	5	10	2	5	10	2	5	10
5×10^{-7}	34.53	34.09	34.90	5.72	5.74	5.35	10.58	10.03	9.51
1×10^{-6}	35.51	34.29	36.28	6.80	6.35	6.86	8.82	7.11	10.32
2×10^{-6}	**40.35**	40.48	42.02	**7.17**	6.95	6.46	**12.09**	13.57	12.25

Regularizing SDPO. Table 13 adds three common stabilizers to SDPO at its main-experiment setting; everything else is unchanged. The first adds, with weight $\beta = 0.05$, the forward KL divergence between the policy and the frozen Base over the full vocabulary at every logged prefix $(x, y_{<t})$, normalized like the main loss. The second adds, with the same weight, the mean over logged tokens of $\log \pi_\theta(y_t \mid x, y_{<t}) - \log \pi_{\text{Base}}(y_t \mid x, y_{<t})$; this term is zero when the policy equals Base but its gradient is not, and it is equivalent to subtracting β from every token advantage. The third winsorizes each token advantage to ± 4.66, the 99th percentile of $|V_t|$ measured on Base, leaving 99% of tokens untouched. Neither KL variant improves SDPO (LC 41.93 and 42.68 against 42.88), and clipping brings LC to 1.74. These regularizers constrain the distance from Base, shift every advantage by a constant, or bound the largest advantages; none of them changes which part of the advantage varies with the message, so the message-independent term identified in Section 4.1 continues to drive the update, and none of the variants approaches CoSD (45.41).

Table 13: SDPO with common regularizers (Qwen3-4B × WildChat, learning rate 2×10^{-6}, 2 epochs).

Variant	LC	AH	AC
SDPO	42.88	6.07	11.63
+ full-vocabulary KL to Base (β=0.05)	41.93	6.49	12.78
+ logged-token log-ratio (β=0.05)	42.68	5.89	13.33
+ advantage winsorizing ($\pm$4.66)	1.74	2.55	1.29

I On-Policy and Off-Policy Experiments: Details and Gemma3-12B Results

Configuration. Data: the TL;DR task from `openai/summarize_from_feedback` (Stiennon et al., 2022). Instruction: "Write a brief summary of the text that begins with 'TL;DR:'." 250 training prompts and 256 held-out evaluation prompts, disjoint. Training: reply cap 258 tokens, prompt cap 1024 tokens, temperature 1.0, learning rate 5×10^{-6}, batch 16, seed 1234 for interaction order, checkpoints at 50/100/150/200/250 interactions. User simulator and judge: both Qwen3-8B, following the setup of Buening et al. (2026), with only the persona description in the system prompt. The judge compares pairs with position swap, discarding ties. "vs Base" is the win rate against the frozen Base of the same backbone. Off-policy setting: 250 responses pre-generated by DeepSeek-V4-Flash-0731, checked for no preamble or markdown that the base model never produces, truncated to the same 258-token cap. Feedback is sampled in tiers according to where the response falls on the persona's preferred dimension: responses at the preferred end receive satisfied feedback, responses at the other end receive requests for change, and responses in between receive a mixture. Reference messages: a fixed pool of 1,950 second-turn user messages from WildChat, shared by all settings.

Personas. "More formatting": the user likes the assistant to use lists and markdown, and the more lists the better. "Commit to one answer": the user wants the assistant to give a single best answer rather than list several options, and dislikes hedging adverbs such as "generally," "typically," "often," and "usually."

Results on Gemma3-12B. Figure 7 repeats Figure 5 on Gemma3-12B. At the final checkpoint CoSD again exceeds both SDPO and Base in all four cells, and SDPO ends below Base in two. On-policy the gap is large for both personas: SDPO stays near zero under "more formatting" and falls back to 30.43 after a peak of 65.12 under "commit to one answer," while CoSD stays above 74 throughout. Off-policy the gap is the smallest of all eight cells: under "more formatting" SDPO is also far above Base (88.41 against 97.35 for CoSD at the final checkpoint), and under "commit to one answer" it nearly closes the gap at 200 interactions before CoSD pulls ahead again (73.21 against 56.25).

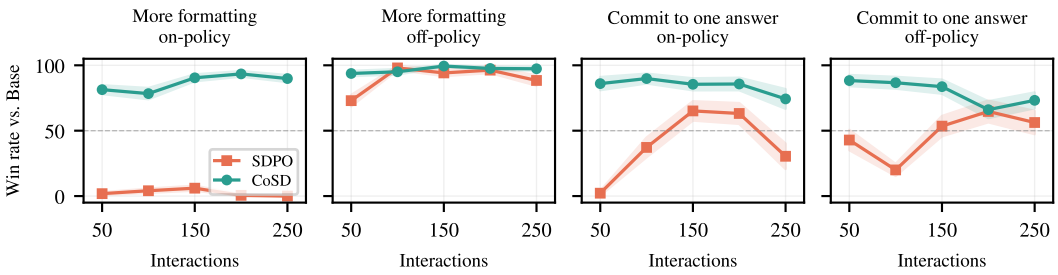

Figure 7: On-policy and off-policy experiments on Gemma3-12B (TL;DR). Win rate vs. Base over 250 interactions. Shading = ± 1 SE. Dashed line = 50%.

J GEMMA3-12B: DETAILED RESULTS AND BACKGROUND MEASUREMENT

Detailed results. Table 14 lists the values plotted in Figure 4.

Table 14: Gemma3-12B results on real-world logs (values of Figure 4). Bold = best in each column and strictly above Base.

Method	LC	AH	AC	IF-s	MMLU-Pro
Base	51.29	7.88	71.28	77.82	59.76
WC SDPO	19.25	1.46	1.74	62.29	44.56
WC CoSD	**52.97**	7.28	**71.35**	75.05	59.42
WF SDPO	35.95	1.78	1.39	66.17	40.58
WF CoSD	**53.83**	**9.07**	**78.41**	**78.56**	**60.24**

Background measurement. This appendix repeats the measurement of Section 3.2 on Gemma3-12B, using the same 2,000 WildChat replies, the same template and rendering, and the same four scoring conditions, with Base weights and no training.

The three readings of Section 3.2 hold. Table 16 and Figure 8 report the same quantities as Figure 1. On Gemma3-12B the reference background reaches 0.70 of the real-message token RMS and the empty template 0.56 (Qwen3-4B: 0.64 and 0.52); a global constant removes only 6.5% of the background mean square (Qwen3-4B: 22%); and the residual after empty-template subtraction is reproducible across two disjoint reference blocks ($r = 0.85$). The decile curves in Figure 8(d) rise steeply in the top decile, which consists of short replies (median 19 tokens).

The background sits on the end-of-turn token. Splitting the scored tokens by category shows that the background is not carried by markdown structure: on both backbones the markdown tokens have the smallest mean background of the four content categories (Table 16, lower block). It is carried instead by the appended end-of-turn token. On Gemma3-12B, conditioning on any message raises the log-probability of this token by 3.56 nats on average (95% CI [3.29, 3.82]; median 1.53; positive on 72% of replies), and the shift under the real follow-up is the same (3.72), so it carries no information about the message. With this token excluded, the signed reply-level background is -0.014 and global centering removes 0.3% of what remains. On Qwen3-4B the end-of-turn shift has median zero and the background is spread over ordinary word tokens. In the top background decile on Gemma3-12B, 56% of the summed background falls on this one token (Qwen3-4B: 8%).

Masking the end-of-turn advantage does not recover SDPO. Since the background on Gemma3-12B peaks at one position, a natural fix is to zero the SDPO advantage there and leave the rest of the update unchanged, so that the end-of-turn token receives no gradient. Table 15 shows that this does not help: 31% of outputs still hit the generation cap, and all five metrics stay at the SDPO level, far below Base. Together with the constant-centering control in Table 10, this rules out the two simple corrections: removing a global constant does not suffice, and removing the largest

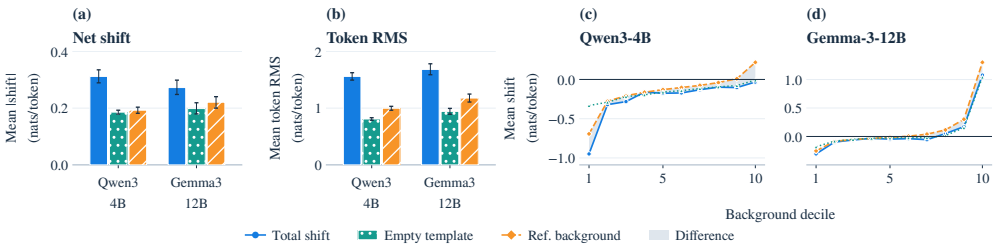

Figure 8: Background measurement on Gemma3-12B, same protocol and panels as Figure 1, with Qwen3-4B re-scored alongside. (a) Reply-level mean absolute shift. (b) Per-reply token RMS. (c,d) Shift by background decile.

single position does not suffice either. What remains after either correction is a background that varies across responses and across positions, which is what the per-token contrast of CoSD removes without having to locate it.

Table 15: Zeroing the SDPO advantage at the end-of-turn position (Gemma3-12B × WildChat). At cap: fraction of AlpacaEval outputs of at least 14,400 characters, about 90% of the 4,096-token generation limit (same measure as Table 5). †: more than 8% of outputs are at cap, so judge scores largely reflect truncated outputs and are not read as quality. Bold = best in each column and strictly above Base.

Method	LC	AH	AC	IFEval	MMLU-Pro	At cap
Base	51.29	7.88	71.28	77.82	59.76	0.0%
SDPO	19.25†	1.46†	1.74†	62.29	44.56	38.3%
SDPO, end-of-turn advantage zeroed	22.05†	2.13†	1.97†	61.37	45.55	31.3%
CoSD	**52.97**	7.28	**71.35**	75.05	59.42	0.1%

K External Reward Models and Log-Based Self-Refinement

This appendix compares CoSD on Qwen3-4B × WildChat with two families of methods that use resources beyond the logs: an external reward model, and UNO (Wang et al., 2026), which learns from user feedback through self-reflection and self-filtering.

External reward model. We test whether an external reward model (Skywork-Reward-V2) adds to the log-only signal. Ranking self-generated responses with the reward model improves DPO and SimPO, but neither reaches CoSD, and training on the union of CoSD data and reward-ranked pairs does not exceed CoSD alone (Figure 9). Adding a reward-model term directly to the CoSD advantage stays within seed noise at small weight and degrades monotonically as the weight grows (Table 17).

Setup. Reward model: Skywork-Reward-V2-Llama-3.1-8B (Liu et al., 2026). RM-DPO and RM-SimPO construct preference pairs by sampling $N{=}4$ responses from Base (temperature 0.8, top-p 0.95) and ranking with the reward model (highest = chosen, lowest = rejected). MIX1 and MIX2 take the union of CoSD training data and RM-DPO preference pairs, applying each row's own loss. MIX1 trains for 1 epoch (766 steps, matching both parent configurations in steps, effective batch, and total row visits). MIX2 trains for 2 epochs.

For advantage-level combination, each token advantage becomes $\widehat{C}_t + \lambda z$, where $z = \mathrm{clip}\big((r_{\mathrm{RM}}(x,y) - m)/(1.4826\,\mathrm{MAD}), -3, 3\big)$, with m the median and MAD the median absolute deviation of r_{RM} over the training set. The added term is a reward-weighted log-likelihood term of the same form as Eq. 7.

RM replacement (weak advantages). Rows are sorted by response length into 20 equal-frequency strata, and within each stratum the sequence RMS of the SDPO token advantages, $\|V\| =$

Table 16: Background on Gemma3-12B versus Qwen3-4B (nats per token). Upper block: the readings of Section 3.2. Middle block: the end-of-turn token. Lower block: mean reference background by token category.

	Qwen3-4B	Gemma3-12B
Reference background / real-message token RMS	0.640	0.700
Empty template / real-message token RMS	0.519	0.560
Mean square remaining after global centering	78.0%	93.5%
Residual repeatability (Pearson r)	0.878	0.850
End-of-turn: mean background shift [95% CI]	+0.40 [+0.31, +0.50]	+3.56 [+3.29, +3.82]
End-of-turn: median shift / fraction positive	0.00 / 48%	+1.53 / 72%
End-of-turn: mean shift under the real follow-up	−0.08	+3.72
Signed reply-level background excluding end-of-turn	−0.159	−0.014
Mean square remaining after centering, excluding end-of-turn	78.8%	99.7%
Top decile: median reply length / share of background on end-of-turn	53 tok / 8%	19 tok / 56%
Markdown structure tokens (6%)	−0.015	−0.016
Newline tokens (1–4%)	−0.103	−0.036
Word tokens (72–76%)	−0.147	−0.024
Punctuation tokens (16–18%)	−0.084	−0.012

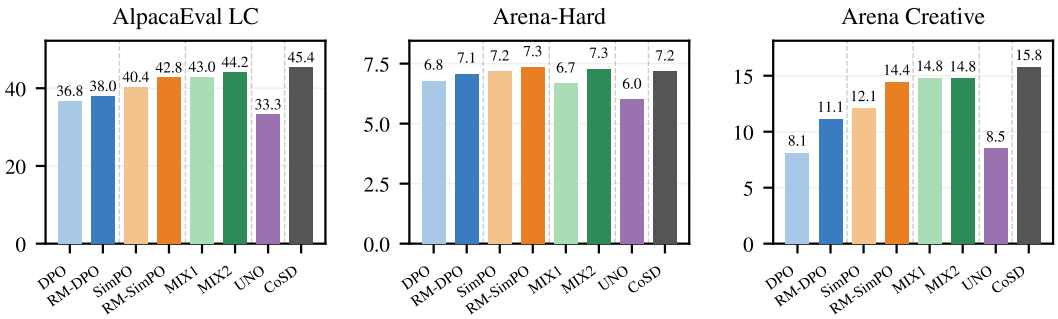

Figure 9: Comparison with external-reward-model methods and UNO (Qwen3-4B × WildChat). RM-: preference pairs ranked by Skywork-Reward-V2. MIX: union of CoSD data and RM-DPO pairs, 1 or 2 epochs. UNO: Wang et al. (2026), clustering threshold 30.

Table 17: Advantage-level reward-model combination. Δ relative to CoSD in Table 1 (45.41).

Setting	LC	Δ vs. CoSD	AH	AC
λ=0.25	45.52	+0.11	7.06	16.44
λ=0.5	43.69	−1.72	6.67	14.85
λ=1.0	42.02	−3.39	6.75	15.10
RM replacement (weak advantages)	46.06	+0.65	7.96	15.29
CoSD initialized from RM-DPO	46.24	+0.83	6.21	14.50

$(\frac{1}{L}\sum_t V_t^2)^{1/2}$ computed with Base, is replaced by its percentile rank. Rows below the median of this percentile (25,000 rows) use z in place of $\widehat{C}_t$; the other 25,003 rows keep $\widehat{C}_t$.

CoSD initialized from RM-DPO. CoSD training starts from the RM-DPO checkpoint, with data and loss unchanged.

UNO. UNO distills user logs into rules and preference pairs, clusters them, and uses a self-judged cognitive gap and a simulated verifier to decide, per cluster, whether to train an expert adapter or a critic adapter that revises the base model's answer at inference time. It is not aligned with our setting: it requires multiple rounds of generation and self-judging during training, and may apply reflection at test time. We follow all hyperparameters of the original paper except the clustering distance threshold, which we raise from 4 to 30, since the original value produces several hundred clusters on WildChat, too many to train an adapter for each. UNO reaches 33.28 LC, 6.00 AH, and 8.49 AC, close to Base and below CoSD on all three metrics (Figure 9).