Aligning Language Models from User Interactions

Anonymous Authors[1]

Abstract

Multi-turn user interactions are among the most abundant data produced by language models, yet we lack effective methods to learn from them. While typically discarded, these interactions often contain useful information: follow-up user messages may indicate that a response was incorrect, failed to follow an instruction, or did not align with the user's preferences. Importantly, language models are already able to make use of this information in context. After observing a user's follow-up, the same model is often able to revise its behavior. We leverage this ability to propose a principled and scalable method for learning directly from user interactions through self-distillation. We condition the model on the user's follow-up message and distill the resulting hindsight token distribution back into the current policy. Remarkably, we show that training on real-world user conversations from WildChat improves language models across standard alignment and instruction-following benchmarks, without regressing other capabilities. The same mechanism enables personalization and continual adaptation, allowing models to learn from and adapt to individual users purely through interaction without any explicit supervision. Our results demonstrate that raw user interactions that arise naturally during deployment can enable alignment, personalization, and continual adaptation at scale.

1 Introduction

In modern language models, inference has overtaken training as the dominant consumer of compute, with models serving massive volumes of user queries every day. Yet the information revealed through these interactions is typically discarded and does not contribute to improving the model itself. At scale, users engage in extended conversations, refining prompts, requesting revisions, and responding directly to model outputs. These interactions are rich with implicit learning signals: follow-up messages may indicate that a response was incorrect, failed to follow an instruction, or did not align with the user's preferences (Don-Yehiya et al., 2024). For example, a user may report an error after executing generated code, point out that a specific instruction was not followed, or ask for a different style or tone. Such signals arise organically during normal use and reflect how model outputs are received and acted upon during deployment. Finding ways to leverage this abundant data source can open the door to continual learning in rich deployment environments at an unprecedented scale.

Despite their scale and richness, we still lack effective methods to learn directly from user interactions. Unlike standard datasets (Ouyang et al., 2022; Chung et al., 2024), user interactions do not come with explicit labels, expert demonstrations, preference comparisons, or rewards. Instead, feedback is implicit and expressed through natural language responses whose meaning depends on the surrounding interaction context. As a result, it is unclear how to train directly on such conversations in a principled manner. At the same time, we observe that language models already demonstrate the ability to leverage this interactive information, a capability known as in-context learning (Brown et al., 2020; Wei et al., 2022). In multi-turn conversations, models often revise their behavior effectively after observing a user's follow-up. In these cases, conditioning on the user's follow-up message leads to responses that are more aligned with the task and the user's intent.

These observations suggest a simple but powerful perspective: having observed the user's follow-up message, i.e., the effect of its action, the model's behavior is often better aligned than before. The user interaction reveals information that the model can already interpret and act upon, but only after the fact. *In hindsight*. Crucially, this improvement arises without additional supervision. This suggests that a model's in-context learning ability can be used as a lever for learning directly from user interactions in a principled way.

Based on this idea, we introduce a simple and scalable method for learning directly from user interactions by comparing a model's original behavior to what it would have

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

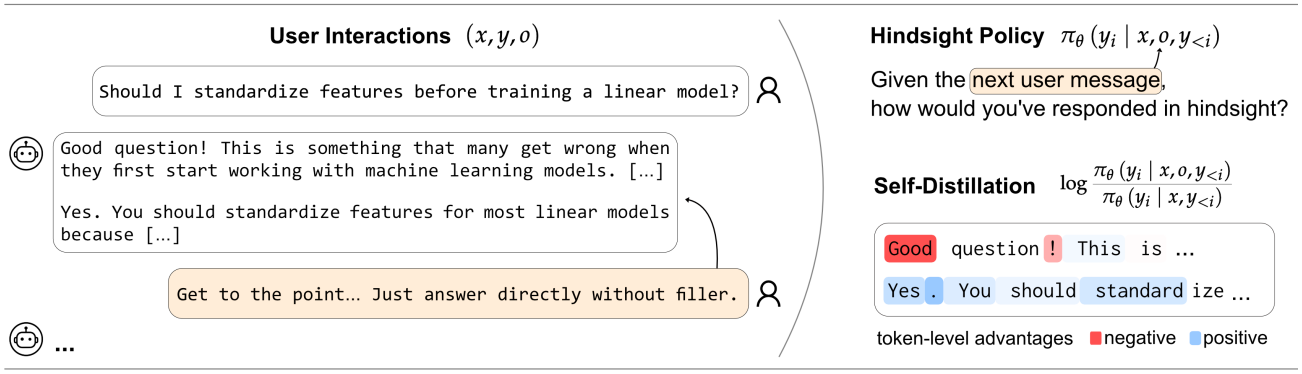


Figure 1. **Direct Learning from User Interactions via Self-Distillation.** From multi-turn user conversations, we obtain several interactions (x, y, o) that consist of the conversation history x, the model's response y, and the subsequent user message o. By conditioning on the user's follow-up, we form the *hindsight policy* and compare it to the original policy, producing token-level advantages that reinforce or penalize parts of the model's original response. In this example, the user's follow-up requests a more direct answer, leading to penalizing filler tokens and reinforcing the actual answer.

done in hindsight. Concretely, after observing a user's follow-up, we reprompt the same model with this additional context and obtain a *hindsight token distribution* that reflects how the model would respond if it had access to the information revealed by the user. By comparing the original policy to this hindsight policy at every token of the original generation, we obtain a comparative learning signal that identifies how the model's behavior should change. We then distill this signal back into the original policy using only the observed interaction. In other words, we distill the model into itself.

Building on recent work on self-distillation (Hübotter et al., 2026), we refer to this approach as Self-Distillation Policy Optimization (SDPO) from User Interactions. We show that this approach (illustrated in Figure 1) is simple and scalable, and enables language models to improve from raw, real-world user conversations without explicit supervision, rewards or preference labels, external models, or additional generations. Remarkably, when applied to real-world user interactions from WildChat (Zhao et al., 2024), this improves alignment and instruction-following performance across standard benchmarks without degrading other capabilities. We also demonstrate that the same self-distillation mechanism naturally supports personalization and continual adaptation, allowing models to adapt to individual users purely through interaction.

2 Problem Formulation

The interaction between a language model and a user consists of a sequence of alternating assistant messages y_t and user messages o_t. At the t-th turn, the language model observes the conversation history $x_t = (o_0, y_1, o_1, \ldots, o_{t-1})$, and generates a response $y_t \sim \pi_\theta(\cdot \mid x_t)$.[1] In turn, the user responds with o_t, assuming the conversation does not terminate. In case of a fresh conversation initiated by the user, the first interaction reduces to the initial prompt o_0, the assistant's answer y_1, and the user's follow-up o_1. We refer to the triple (x_t, y_t, o_t) as a single interaction. Consequently, a user conversation $(o_0, y_1, \ldots, y_t, o_t)$ yields t interactions $(x_1, y_1, o_1), \ldots, (x_t, y_t, o_t)$. We use (x, y, o) to denote a generic single interaction.

Despite the ubiquity of conversational data of this form, it remains unclear how to leverage user interactions directly. One could attempt to introduce auxiliary mechanisms, such as semantic categorization of conversations (Shi et al., 2024; Gunjal et al., 2025), explicit preference annotation (Stephan et al., 2024; Lee et al., 2024), or other post-hoc extracted rewards (Wang et al., 2026; Urcelay et al., 2026), but even then it is not obvious how to construct such signals reliably from raw interaction data alone. Any such approach would require additional modeling assumptions and intermediate objectives that are external to the interaction itself. As a result, they do not provide a simple or principled way of training models from user interactions as they naturally occur. We discuss related work in Appendix A.

Accordingly, we define the problem of learning directly from multi-turn conversations without other externalities as **Direct Learning from User Interactions**. This motivates the central question of this work:

> Can we train language models *directly* from multi-turn user interactions in a simple, principled, and scalable manner?

More specifically, can we both improve general alignment capabilities and enable continual personalization to users from interactions alone without explicit supervision?

[1] In practice, the assistant may condition on a representation of x_t, such as a sliding context window or a summary. For simplicity, we treat x_t as the full interaction history here.

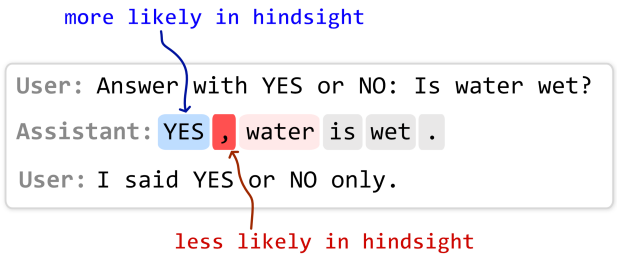


Figure 2. Example of the *token-level advantages* from Equation (1). Positive advantages are in blue and negative ones in red.

3 Direct Learning from User Interactions

Naturally occurring user interactions often contain implicit signals about the adequacy of an assistant's response. Follow-up user messages may indicate that a response was incorrect, failed to follow an instruction, or did not align with the user's preferences, even when no explicit feedback is provided.

Leveraging In-Context Capabilities. Modern language models are often able to make effective use of such information in context: when conditioned on a follow-up message such as an error report, a clarification, or a requested revision, the model can frequently produce outputs that correct previous mistakes, better satisfy constraints, or more closely match the user's preferences. In other words, *in hindsight*, the model's distribution is better aligned to the task.

We leverage this capability by considering the *hindsight* distribution $\pi_\theta(\cdot \mid x, o)$, which conditions not only on the interaction history x but also on the observed user continuation o through reprompting the original policy with x and o (cf. the prompting template in Table 1). The resulting token distribution reflects how the model would respond if it were given access to the additional information revealed by the next user message. Empirically and intuitively, $\pi_\theta(\cdot \mid x, o)$ is often better aligned with the task at hand than the original policy $\pi_\theta(\cdot \mid x)$.

This perspective admits a fine-grained, token-level interpretation. Letting y_i be the i-th token of the completion y generated from $\pi_\theta(\cdot \mid x)$, we can compare the token probabilities $\pi_\theta(y_i \mid x, y_{<i})$ and $\pi_\theta(y_i \mid x, o, y_{<i})$. When the hindsight model $\pi(\cdot \mid x, o)$ assigns lower probability to a particular token y_i, this indicates that the user's response provides evidence that this token (or the trajectory it induces) contributed to an undesirable outcome. Conversely, tokens whose likelihood increases under $\pi_\theta(\cdot \mid x, o)$ are reinforced by the user's response. The resulting log-ratio (i.e., log-difference) $\log \pi_\theta(y_i \mid x, o, y_{<i}) - \log \pi_\theta(y_i \mid x, y_{<i})$ can thus act as a comparative learning signal from user interactions, and will serve as the fundamental learning signal throughout the paper. This now admits two equivalent views.

Policy Gradient. One useful way to interpret the log-ratio is as a *token-level advantage*

$$A_i(x, y, o) := \log \frac{\pi_\theta(y_i \mid x, o, y_{<i})}{\pi_\theta(y_i \mid x, y_{<i})}, \tag{1}$$

which measures how the likelihood of a token changes after conditioning on the user's response. Using this advantage in a standard policy gradient update reinforces tokens whose probability increases under the hindsight distribution and penalizes those whose probability decreases. We will refer to the advantage interchangeably as the token-level advantage or the SDPO advantage. Figure 2 provides an illustrative example. In this view, learning corresponds to increasing the log-ratio in expectation, treating it as a fixed advantage per update step that is not differentiated w.r.t. θ.

Self-Distillation. Equivalently, and perhaps more conveniently from an optimization perspective, we can update $\pi_\theta(\cdot \mid x)$ to more closely match the hindsight policy $\pi_\theta(\cdot \mid x, o)$ by minimizing the reverse KL divergence. Here, the hindsight policy acts as a teacher and is treated as a fixed target during each update, for which we define the detached hindsight model $\overline{\pi}_\theta(\cdot \mid x, o) := \text{stopgrad}(\pi_\theta(\cdot \mid x, o))$. We first sample $y \sim \pi_\theta(\cdot \mid x)$ and then minimize a standard distillation loss,

$$\mathcal{L}_{\text{SDPO}}(\theta) := \sum_i \text{KL}\big(\pi_\theta(\cdot \mid x, y_{<i}) \,||\, \overline{\pi}_\theta(\cdot \mid x, o, y_{<i})\big), \tag{2}$$

As shown in Hübotter et al. (2026), the gradient is

$$\nabla_\theta \mathcal{L}_{\text{SDPO}}(\theta) = - \mathbb{E}_{y \sim \pi_\theta}\left[\sum_i \mathbb{E}_{y_i \sim \pi_\theta}\Big[\nabla_\theta \log \pi_\theta(y_i \mid x, y_{<i})\, A_i(x, y, o)\Big]\right] \tag{3}$$

Interestingly, Lemma C.1 in Appendix C demonstrates that the policy gradient with token-level advantages is an unbiased one-sample approximation of the self-distillation gradient. Hence, the two perspectives yield equivalent gradient updates in expectation, differing only in whether the log-ratio is interpreted as an advantage or as a distillation loss. In our experiments, we adopt this policy gradient perspective for its simplicity.

Following recent work on self-distillation (Hübotter et al., 2026), we refer to the corresponding algorithm as Self-Distillation Policy Optimization (SDPO) from User Interactions. Algorithm 1 outlines the approach for learning from online user interactions, where an update is performed after observing the user's next message. In practice, interaction data is often available as logged conversations, possibly with completions generated from a different model. To this end, it is also natural to consider an offline and off-policy variant of SDPO, which we discuss in Section 4.1.

Algorithm 1 Self-Distillation Policy Optimization from User Interactions

input: language model π_θ
repeat
  observe context x_t including the most recent user message o_{t-1}
  sample answer $y_t \sim \pi_\theta(\cdot \mid x_t)$ with log-probabilities $\log \pi_\theta(y_{t,i} \mid x_t, y_{t,<i})$
  observe user message o_t in response to y_t assuming the conversation does not terminate
  compute token log-probabilities of hindsight policy $\log \pi_\theta(y_{t,i} \mid x_t, o_t, y_{t,<i})$
  update current model π_θ with gradient $\nabla_\theta \mathcal{L}_{\text{SDPO}}(\theta)$
until converged

User:	`<conversation history including most recent user prompt>` x
	`<hindsight context>` The following is a future user message. Use this to guide your answer to the user prompt: o
Assistant:	`<assistant completion>` y

Table 1. Template for the hindsight policy $\pi_\theta(y \mid x, o)$. We recover the template for the base policy $\pi_\theta(y \mid x)$ when removing "`<hindsight context>` [...]" from the user prompt.

Self-Distillation as an Alignment Objective. A common framing of alignment is that a language model should maximize a user's latent reward function $r(x, y)$, which is unobserved and difficult to specify or estimate in practice. Since SDPO learns directly from naturally occurring user interactions, it is not immediately obvious how it relates to this traditional reward-maximization view of alignment. Under a stylized model of user behavior and language model conditioning, we find that SDPO admits a simple and intuitive interpretation as implicitly optimizing the latent reward of the interacting user.

Proposition 3.1 (Informal, Appendix B)**.** *Under idealized assumptions on user responses and model conditioning, the sequence-level self-distillation advantage satisfies*

$$\log \frac{\pi_\theta(y \mid x, o)}{\pi_\theta(y \mid x)} = r(x, y) - \log Z(x, y),$$

where $Z(x, y)$ is a normalization term. In other words, under idealized assumptions, SDPO can be interpreted as implicitly maximizing the interacting user's latent reward.

4 Experimental Results

We evaluate SDPO from User Interactions with respect to two central questions:

1. **General Alignment:** Can learning directly from raw, real-world user conversations improve the general alignment and instruction-following capabilities of language models?
2. **Continual Personalization and Adaptation:** Can we continually personalize language models from online user conversations without any explicit supervision?

Section 4.1 addresses the first question. We train SDPO on offline user conversations from WildChat (Zhao et al., 2024) and WildFeedback (Shi et al., 2024). These datasets consist of real-world user interactions and contain no explicit supervision signals. We evaluate the resulting models on standard alignment, instruction-following, math and coding, and knowledge tasks. Remarkably, we show that training on real-world user conversations with SDPO improves alignment and instruction-following, without regressing other capabilities.

Section 4.2 addresses the second question. We demonstrate that SDPO enables continual personalization through interaction by simulating users with distinct preferences and evaluating the model's ability to adapt to these preferences over time from interactions alone. We show that SDPO can adapt online to changing user preferences and does not forget previously learned behavior.

Finally, Section 4.3 qualitatively analyzes the self-distillation advantages from Equation (1) at illustrative user interactions. We find that the learning signal is highly interpretable and is robust to irrelevant next user messages, e.g., when the user changes topics mid-conversation.

4.1 General Alignment from Real-World User Conversations

We train SDPO on user conversations from WildChat (Zhao et al., 2024) and WildFeedback (Shi et al., 2024), a curated subset of WildChat containing approximately 20,000 individual conversations. Around 6,000 of these consist only of a single prompt-response pair and therefore do not contain a user follow-up.

We evaluate SDPO across two model families and four

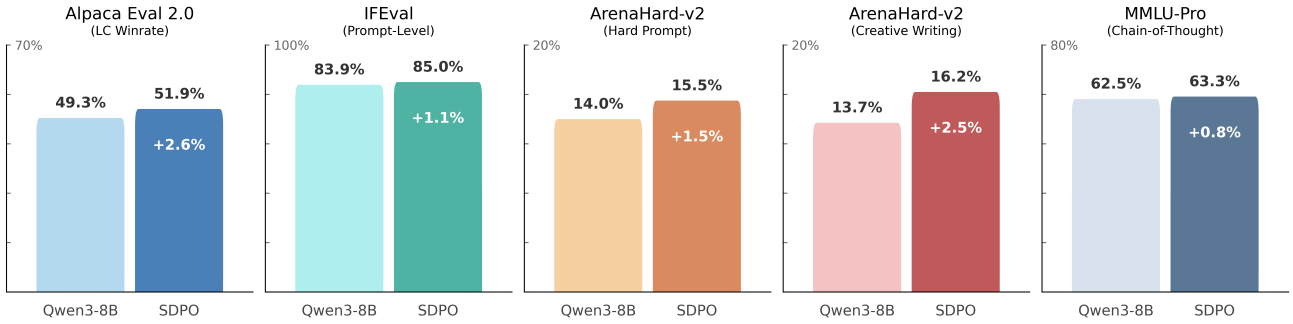


Figure 3. **Training on real-world user conversations, SDPO improves general alignment and instruction-following performance across benchmarks, without regressing other capabilities.** Results for Qwen3-8B before and after training on 14,000 conversations.

	Alpaca Eval 2.0 (LC Winrate)	**IFEval** (Prompt-Level)	**ArenaHard-v2** (Hard Prompt)	**ArenaHard-v2** (Creative Writing)	**MMLU-Pro** (Chain-of-Thought)
Qwen3-4B	37.9	81.9	9.0	8.0	58.1
SDPO	↑ **46.1**	↑ **83.2**	↓ **7.8**	7.9	58.0
Qwen3-8B	49.3	83.9	14.0	13.7	62.5
SDPO	↑ **51.9**	↑ **85.0**	↑ **15.5**	↑ **16.2**	↑ **63.3**
Olmo3-7B-SFT	34.3	80.2	2.4	1.4	23.7
SDPO	↑ **35.2**	↑ **80.6**	2.4	1.4	↑ **24.0**
Olmo3-7B-DPO	50.4	80.2	1.7	8.2	28.4
SDPO	↑ **51.8**	↑ **80.4**	↑ **2.0**	↑ **10.0**	↑ **28.7**

Table 2. **Across model families and model sizes, SDPO improves alignment and instruction-following without degrading other capabilities.** A mild exception is Qwen3-4B, where SDPO significantly increases performance on AlpacaEval 2.0 (+8.2%) and IFEval (+1.3%) but decreases performance on the math and coding tasks of ArenaHard-v2 (-1.2%). We only show arrows when performance changed by more than 0.1 percentage points.

models overall. Specifically, we use Qwen3-4B and Qwen3-8B (Qwen Team, 2025), as well as Olmo3-7B-Instruct-SFT and Olmo3-7B-Instruct-DPO, which are the SFT and DPO checkpoints from the Olmo3 model family (Olmo et al., 2025). All models are trained using the same 14,000 user coversations and evaluated using identical benchmark protocols. We evaluate each model before and after SDPO training on AlpacaEval 2.0 (Dubois et al., 2024), IFEval (Zhou et al., 2023), ArenaHard-v2 (Li et al., 2025; 2024), and MMLU-Pro (Wang et al., 2024a) to cover alignment, instruction-following, math and coding, creative writing, and knowledge tasks. We provide additional experimental details in Appendix D.

Off-Policy SDPO from Logged User Interactions. As the assistant completions in WildChat were generated GPT-3.5 Turbo and GPT-4 the interactions are off-policy. In principle, unbiased off-policy policy gradient updates would require access to the behavioral policy or its token-level probabilities, which are not available for these datasets. Instead, we optimize a surrogate SDPO objective defined directly over the logged interactions $(x, y, o) \sim \mathcal{D}$:

$$\widehat{\mathcal{L}}_{\text{SDPO}}(\theta) \tag{4}$$
$$= \mathbb{E}_{(x,y,o)\sim\mathcal{D}}\left[\sum_i \mathrm{KL}\big(\pi_\theta(\cdot \mid x, y_{<i}) \,||\, \overline{\pi}_\theta(\cdot \mid x, o, y_{<i})\big)\right].$$

While this objective is biased with respect to the on-policy SDPO loss, it can be interpreted as an off-policy approximation of the SDPO objective.

Main Results. Figure 3 reports the performance of Qwen3-8B before and after training with SDPO. Training on raw, real-world user conversations consistently improves performance on all evaluated tasks. Importantly, we observe no degradation on any benchmark, despite the fact that the training data consists of noisy user conversations. Table 2 summarizes results across all models and benchmarks. For the Olmo3-7B models SDPO yields consistent but often modest improvements. In contrast, Qwen3-4B exhibits a clear trade-off: while SDPO substantially improves performance on AlpacaEval 2.0 (+8.2%), it also leads to a mild decrease (-1.2%) on the math and coding tasks.

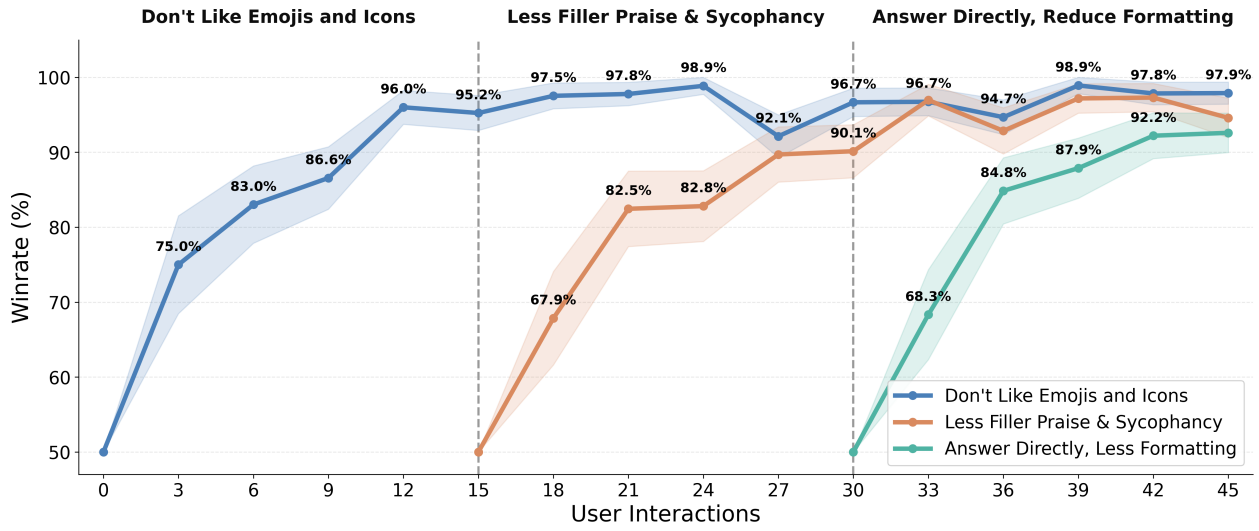


Figure 4. **SDPO enables continual personalization without catastrophic forgetting.** We train a single Qwen3-8B model online with SDPO for 45 user interactions, during which three complementary user preferences are introduced sequentially. Each curve reports the win rate of the current model against the model checkpoint at the time the corresponding preference was introduced, hence isolating the relative improvement along that specific preference dimension. Earlier preferences remain strong as new ones are learned, showing that SDPO can accumulate complementary preferences over time without forgetting previously learned behavior. Shaded regions indicate standard error over 100 held-out prompts.

Overall, these results suggest that SDPO is most effective when the base model can reliably interpret and exploit the hindsight signal provided by user follow-ups. For smaller or less instruction-tuned models, this signal appears weaker or less stable, leading to smaller gains and, in some cases, task-specific trade-offs.

How important is the quality of user conversations? WildFeedback retains roughly 3% of the original conversations in WildChat by coarsely filtering for conversations that contain implicit feedback signals, such as expressions of dissatisfaction, requests for correction, or revision prompts (Shi et al., 2024). In practice, user conversations are abundant so that filtering down to a smaller subset is not a concern. Nevertheless, we evaluate whether SDPO continues to behave sensibly when trained on fully uncurated user interactions. To this end, we train SDPO on a randomly sampled subset of WildChat that matches WildFeedback in scale. All other training and evaluation settings are kept identical.

Table 3 shows that even though filtering for feedback-rich conversations strengthens the learning signal, SDPO is surprisingly robust to the data quality. Even when trained on fully uncurated conversations, SDPO can extract useful alignment signals from user interactions without collapsing performance.

4.2 Continual Personalization and Adaptation from User Interactions

Because SDPO learns directly from user interactions, it naturally enables personalization from those conversations. We evaluate this capability in two complementary experimental settings, where the model interacts with a simulated user without any explicit supervision.

Continual Personalization. In the first setting, we consider complex and complementary user preferences on a broad set of real-world requests from HelpSteer2 (Wang et al., 2024b). We define preference profiles, such as disliking emojis and filler praise (e.g., "Good Question!"), and prompt Qwen3-32B with these user profiles to generate user responses. Given a prompt x from HelpSteer2, the model generates a completion y, and the user simulator responds

	AlpacaEval 2.0 (LC Winrate)	**IFEval** (Prompt-Level)	**ArenaHard-v2** (Hard Prompt)	**ArenaHard-v2** (Creative Writing)	**MMLU-Pro** (Chain-of-Thought)
Qwen3-8B	49.3	83.9	14.0	13.7	62.5
SDPO (WildChat)	↑ **50.7**	↑ **84.5**	↓ **13.4**	↑ **14.0**	62.4

Table 3. **Even on fully uncurated user interactions, SDPO still yields improvements in alignment and instruction-following and only mild degradation in math and coding.** Results for training on a randomly sampled subset of 14,000 conversations from WildChat.

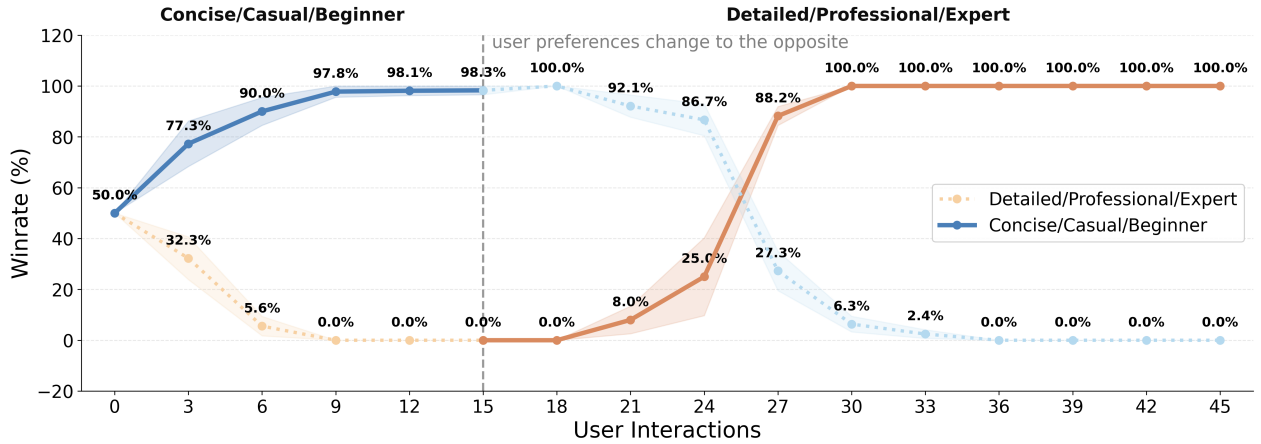


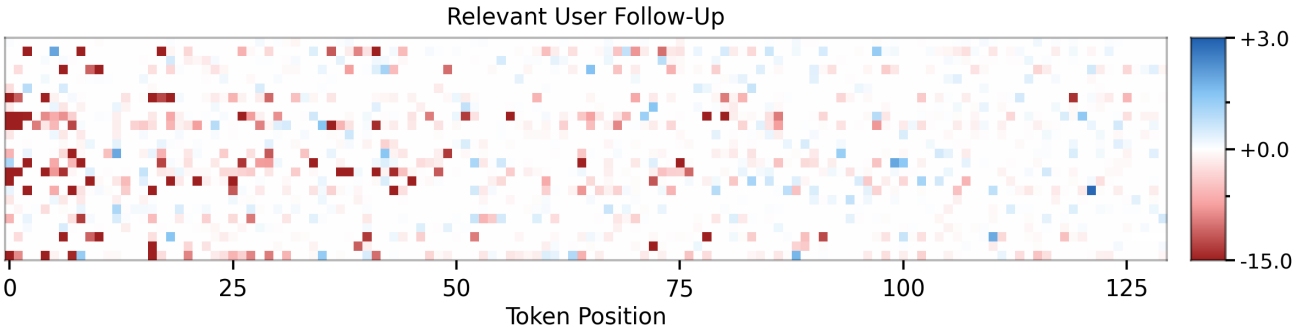
Figure 5. **SDPO adapts online to changing user preferences.** The user's preference about how the model should respond is flipped to its opposite after the first 15 interactions. SDPO with Qwen3-8B is able to quickly reverse the learned behavior. The win rates for the two user profiles are computed against the base model at step 0 over 100 held-out prompts.

with a follow-up o. We train Qwen3-8B on these interactions with SDPO and evaluate the policy against its base version, where we use Qwen3-32B to judge the models on 100 test prompts. We here train fully online with batch size 1, and provide more experimental details in Appendix D.

Figure 4 shows that SDPO quickly adapts the model to the user's preferences (e.g., not using emojis), achieving 75% win rate after only three interactions. Moreover, we are able to incorporate new preferences while retaining previously inferred ones, illustrating that continual personalization does not require forgetting earlier behavior when preferences are compatible.

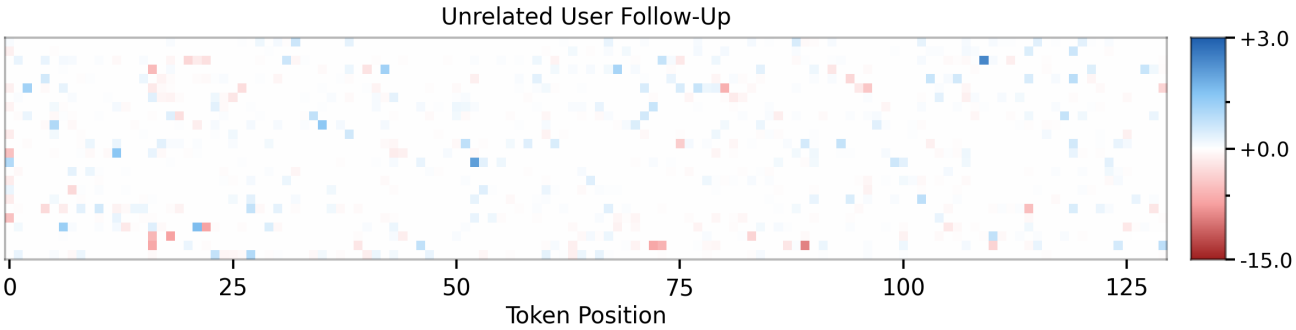
Adapting to Changing User Preferences. In the second setting, we study stylistic personalization in a summarization task using prompts from the tl;dr dataset (Stiennon et al., 2020). We define user-specific writing-style preferences and train Qwen3-8B with SDPO. We again prompt Qwen3-32B with these user profiles to generate user responses and the judge evaluations.

In Figure 5, the model first interacts with a user who prefers concise, casual, and beginner-friendly responses. Then, the user's preferences are abruptly flipped to its opposite (e.g., from concise to detailed). We observe that SDPO quickly adapts to the initial preferences within three interactions, achieving over 75% win rate after only three user interactions. At the same time, we observe rapid adaptation to the change in user preferences, reversing the previously learned behavior within a handful of interactions and converging to the updated preferences after twelve interactions. Additional results are provided in Appendix F.




```
User: Write a short, friendly email to a teammate summarizing project status. Say progress is ...
Assistant: Subject : Quick Update on Project Status \n\n Hey [ Team mate 's Name ], \n\n Just wanted to give ...
User: Rewrite in a formal, professional tone.
```

Figure 6. **When user follow-ups are relevant to the model's completion, we observe strong positive and negative SDPO advantages.** We visualize the advantages with Qwen3-8B for user follow-ups that carry relevant information about the model's answer. **Below:** Example (second line in the heatmap), where the user requests a more formal rewrite of the assistant's draft. Informal expressions (e.g., "Hey") have large negative advantages.




```
User: Write a short, friendly email to a teammate summarizing project status. Say progress is ...
Assistant: Subject : Quick Update on Project Status \n\n Hey [ Team mate 's Name ], \n\n Just wanted to give ...
User: What is 27*4?
```

Figure 7. **When user follow-ups are unrelated to the model's response, SDPO advantages are close to zero.** We visualize the advantages with Qwen3-8B for user follow-ups that are unrelated to the model's generation. **Below:** Following the request to write an email, the user responds with *"What is 27×4?"*. The advantages are close to zero everywhere, which means that SDPO does not meaningfully update the policy from the interaction.

4.3 Interpretability and Robustness of Self-Distillation Advantages

We complement our quantitative results with a qualitative analysis of the learning signal, and visualize the self-distillation advantages $A_i(x, y, o)$ using heatmaps for illustrative user interactions. We compute the advantages with Qwen3-8B for 24 interactions where the next user message is relevant to the model's previous completion and where it is unrelated. Positive advantages (shown in blue) correspond to tokens reinforced by SDPO, while negative advantages (shown in red) correspond to tokens that are penalized.

When user follow-ups provide relevant feedback, such as requests for revision, corrections, or explicit preference statements, we observe strong positive and negative advantages (Figure 6). In contrast, when user follow-ups are unrelated to the model's previous output, the resulting SDPO advantages are close to zero (Figure 7). In these cases, the hindsight policy assigns probabilities similar to those of the original policy, leading to little or no policy update.

Overall, these visualizations highlight two key properties of our self-distillation approach. First, the token-level advantages are highly interpretable and align with intuitive notions of user feedback when such feedback is present. Second, SDPO is robust to irrelevant or uninformative user follow-ups, naturally suppressing learning updates when the interaction does not convey actionable information about the preceding model output.

5 Discussion

We introduced a simple and scalable self-distillation approach for learning directly from user interactions. We leverage the language model's in-context learning capabilities by treating the user's next message as hindsight information, yielding an interpretable token-level learning signal without requiring other auxiliary mechanisms. Empirically, we showed that SDPO improves general alignment and instruction-following performance when trained on raw, real-world user conversations, supports continual personalization from interaction alone, and remains robust to noisy, uncurated, or irrelevant user follow-ups.

More broadly, our results highlight naturally occurring user interactions as a distinct and underutilized data modality for improving deployed language models. Unlike traditional training data, user conversations arise naturally during deployment and reflect how model outputs are actually used, evaluated, and acted upon in real-world settings and applications. The scale and diversity of such data far exceed that of manually curated datasets, suggesting substantial potential for learning systems that close the loop between deployment and training. Our findings indicate that even simple, local learning signals extracted from user interactions can be sufficient to drive meaningful adaptation.

Ethics Statement

Learning directly from user interactions introduces important safety and ethical considerations. User follow-ups may implicitly encourage behaviors that conflict with existing safety or alignment constraints, for example by rewarding evasive, misleading, or policy-violating responses through repeated interaction. In particular, continual personalization without additional guardrails raises risks that adaptive updates could be exploited by users attempting to steer the model toward unsafe or manipulative behavior over time. While SDPO from User Interactions derives a local, token-level learning signal and naturally suppresses updates

from irrelevant interactions, it does not by itself distinguish between benign and adversarial learning signals. Nevertheless, the hindsight prompt may offer the ability to endow the model with principles based on which to act and interpret user feedback. More broadly, the collection and use of user interaction data for learning must be accompanied by appropriate transparency, consent, and governance mechanisms.

References

Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Garea, S. R., Geist, M., and Bachem, O. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Auzina, I. A., Strüber, J., Hernández-Gutiérrez, S., Goel, S., Prabhu, A., and Bethge, M. Intrinsic credit assignment for long horizon interaction, 2026.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. In *NeurIPS*, 2020.

Cao, B., Cai, D., and Lam, W. Infiniteicl: Breaking the limit of context window size via long short-term memory transformation. In *ACL*, 2025.

Chen, A., Scheurer, J., Campos, J. A., Korbak, T., Chan, J. S., Bowman, S. R., Cho, K., and Perez, E. Learning from natural language feedback. *TMLR*, 2024.

Chen, J. C.-Y., Peng, B. X., Choubey, P. K., Huang, K.-H., Zhang, J., Bansal, M., and Wu, C.-S. Nudging the boundaries of llm reasoning. In *ICLR*, 2026.

Chen, W., Chen, J., Tajwar, F., Zhu, H., Duan, X., Salakhutdinov, R., and Schneider, J. Retrospective in-context learning for temporal credit assignment with large language models. In *NeurIPS*, 2025.

Choi, E., Jo, Y., Jang, J., and Seo, M. Prompt injection: Parameterization of fixed inputs. *arXiv preprint arXiv:2206.11349*, 2022.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *JMLR*, 25(70):1–53, 2024.

Don-Yehiya, S., Choshen, L., and Abend, O. Naturally occurring feedback is common, extractable and useful. *arXiv preprint arXiv:2407.10944*, 2024.

Don-Yehiya, S., Burtenshaw, B., Fernandez Astudillo, R., Osborne, C., Jaiswal, M., Kuo, T.-S., Zhao, W., Shenfeld, I., Peng, A., Yurochkin, M., et al. The future of open human feedback. *Nature Machine Intelligence*, 7(6):825–835, 2025.

Dou, Z.-Y., Yang, C.-F., Wu, X., Chang, K.-W., and Peng, N. Re-rest: Reflection-reinforced self-training for language agents. In *EMNLP*, 2024.

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled alpacaeval: A simple way to debias automatic evaluators. In *COLM*, 2024.

Eyuboglu, S., Ehrlich, R., Arora, S., Guha, N., Zinsley, D., Liu, E., Tennien, W., Rudra, A., Zou, J., Mirhoseini, A., et al. Cartridges: Lightweight and general-purpose long context representations via self-study. In *ICLR*, 2026.

Goyal, P., Niekum, S., and Mooney, R. J. Using natural language for reward shaping in reinforcement learning. In *IJCAI*, 2019.

Gunjal, A., Wang, A., Lau, E., Nath, V., He, Y., Liu, B., and Hendryx, S. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Hatamizadeh, A., Prabhumoye, S., Gitman, I., Lu, X., Han, S., Ping, W., Choi, Y., and Kautz, J. igrpo: Self-feedback-driven llm reasoning. *arXiv preprint arXiv:2602.09000*, 2026.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hübotter, J., Lübeck, F., Behric, L., Baumann, A., Bagatella, M., Marta, D., Hakimi, I., Shenfeld, I., Buening, T. K., Guestrin, C., et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Jin, C., Xu, J., Liu, B., Tao, L., Golovneva, O., Shu, T., Zhao, W., Li, X., and Weston, J. The era of real-world human interaction: Rl from user conversations. *arXiv preprint arXiv:2509.25137*, 2025.

Kimi Team, Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kujanpää, K., Marttinen, P., Valpola, H., and Ilin, A. Efficient knowledge injection in LLMs via self-distillation. *TMLR*, 2025.

Lee, K., Hwang, D., Park, S., Jang, Y., and Lee, M. Reinforcement learning from reflective feedback (rlrf): Aligning and improving llms via fine-grained self-reflection. *arXiv preprint arXiv:2403.14238*, 2024.

Li, T., Chiang, W.-L., Frick, E., Dunlap, L., Zhu, B., Gonzalez, J. E., and Stoica, I. From live data to high-quality benchmarks: The arena-hard pipeline, 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard.

Li, T., Chiang, W.-L., Frick, E., Dunlap, L., Wu, T., Zhu, B., Gonzalez, J. E., and Stoica, I. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *ICML*, 2025.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL*, 2022.

Lu, K. and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. URL https://thinkingmachines.ai/blog/on-policy-distillation.

Luo, R., Liu, Z., Liu, X., Du, C., Lin, M., Chen, W., Lu, W., and Pang, T. Language models can learn from verbal feedback without scalar rewards. *arXiv preprint arXiv:2509.22638*, 2025.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

Mitra, P. and Ulukus, S. Semantic soft bootstrapping: Long context reasoning in llms without reinforcement learning. *arXiv preprint arXiv:2512.05105*, 2025.

Olmo, T., Ettinger, A., Bertsch, A., Kuehl, B., Graham, D., Heineman, D., Groeneveld, D., Brahman, F., Timbers, F., Ivison, H., et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Penaloza, E., Vattikonda, D., Gontier, N., Lacoste, A., Charlin, L., and Caccia, M. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Qu, Y., Setlur, A., Smith, V., Salakhutdinov, R., and Kumar, A. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

Scheurer, J., Campos, J. A., Korbak, T., Chan, J. S., Chen, A., Cho, K., and Perez, E. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.

Shao, R., Asai, A., Shen, S. Z., Ivison, H., Kishore, V., Zhuo, J., Zhao, X., Park, M., Finlayson, S. G., Sontag, D., et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Shenfeld, I., Damani, M., Hübotter, J., and Agrawal, P. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Shi, T., Wang, Z., Yang, L., Lin, Y.-C., He, Z., Wan, M., Zhou, P., Jauhar, S., Chen, S., Xia, S., et al. Wildfeedback: Aligning llms with in-situ user interactions and feedback. *arXiv preprint arXiv:2408.15549*, 2024.

Shi, T., Chen, S., Jiang, B., Song, L., Yang, L., and Zhao, J. Experiential reinforcement learning. *arXiv preprint arXiv:2602.13949*, 2026.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Snell, C., Klein, D., and Zhong, R. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Song, Y., Chen, L., Tajwar, F., Munos, R., Pathak, D., Bagnell, J. A., Singh, A., and Zanette, A. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

Stephan, M., Khazatsky, A., Mitchell, E., Chen, A. S., Hsu, S., Sharma, A., and Finn, C. Rlvf: Learning from verbal feedback without overgeneralization. In *ICML*, 2024.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. Learning to summarize from human feedback. In *NeurIPS*, 2020.

Talmor, A., Herzig, J., Lourie, N., and Berant, J. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *NAACL*, 2019.

Urcelay, B. M., Krause, A., and Ramponi, G. From words to rewards: Leveraging natural language for reinforcement learning. In *TMLR*, 2026.

Wang, H., Wang, L., Zhang, C., Mao, T., Qin, S., Lin, Q., Rajmohan, S., and Zhang, D. Text2grad: Reinforcement learning from natural language feedback. In *ICLR*, 2026.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024a.

Wang, Z., Dong, Y., Delalleau, O., Zeng, J., Shen, G., Egert, D., Zhang, J., Sreedhar, M. N., and Kuchaiev, O. Helpsteer 2: Open-source dataset for training top-performing reward models. In *NeurIPS*, 2024b.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. Text2reward: Reward shaping with language models for reinforcement learning. In *ICLR*, 2024.

Yadkori, Y. A., Kuzborskij, I., György, A., and Szepesvári, C. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. In *NeurIPS*, 2024.

Yang, W., Lin, Y., Zhou, J., and Wen, J.-R. Distilling rule-based knowledge into large language models. In *COLING*, 2025.

Yang, Z., Pang, T., Feng, H., Wang, H., Chen, W., Zhu, M., and Liu, Q. Self-distillation bridges distribution gap in language model fine-tuning. In *ACL*, 2024.

Yao, W., Heinecke, S., Niebles, J. C., Liu, Z., Feng, Y., Xue, L., Murthy, R., Chen, Z., Zhang, J., Arpit, D., et al. Retroformer: Retrospective large language agents with policy gradient optimization. In *ICLR*, 2024.

Ye, T., Dong, L., Wu, X., Huang, S., and Wei, F. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Lu, P., Huang, Z., Guestrin, C., and Zou, J. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

Zhao, S., Xie, Z., Liu, M., Huang, J., Pang, G., Chen, F., and Grover, A. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Zhao, W., Ren, X., Hessel, J., Cardie, C., Choi, Y., and Deng, Y. Wildchat: 1m chatgpt interaction logs in the wild. In *ICLR*, 2024.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Zhou, R., Li, S., Zhang, A., and Leqi, L. Expo: Unlocking hard reasoning with self-explanation-guided reinforcement learning. In *NeurIPS*, 2025.

A Related Work

Preference-Based Alignment. Much of recent progress in aligning language models comes from supervised instruction tuning and preference-based post-training, where explicit human or AI feedback is collected as rankings or rewards and optimized via RLHF or direct preference optimization (Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023). These approaches are effective, but they rely on curated datasets that provide explicit feedback for each generation. In contrast, we leverage implicit feedback within real-world user conversations. Though such conversations are abundant, few open datasets of such user conversations exist (Don-Yehiya et al., 2025), since the community has lacked an effective method for learning from them.

Learning from Natural Language Feedback and through Retrospection. Substantial research has focused on translating verbal feedback into reward functions for RL, for example, by mapping feedback to discrete token-level rewards using an external frozen model (Wang et al., 2026) or by using strong external LLMs to explicitly construct state-wise reward functions (Goyal et al., 2019; Xie et al., 2024; Urcelay et al., 2026; Jin et al., 2025). A recent simplified instantiation of this approach has been to manually design so-called rubrics according to which an LLM judge scores generations (Gunjal et al., 2025; Shao et al., 2025; Kimi Team et al., 2025).

Alternatively, feedback can be utilized without explicit reward modeling. Recent research explored in-context improvement without updating model weights (Madaan et al., 2023; Shinn et al., 2023; Yao et al., 2024; Yuksekgonul et al., 2025). Other works manually curate preference datasets by pairing responses before and after feedback to train with direct preference optimization (Stephan et al., 2024; Lee et al., 2024). Chen et al. (2024) perform SFT on refined generations that incorporate feedback. Our approach differs from these works in performing direct credit assignment over the initial model's rollouts without additional generation. In concurrent work, (Auzina et al., 2026) use a related idea to self-distillation for learning how to elicit information from multi-turn conversations by assigning turn-level implicit rewards.

Self-Distillation. Distillation is a general technique for transferring knowledge from a strong teacher model to a student model by mimicking the teacher's output distribution or intermediate representations (Hinton et al., 2015; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Based on this idea, (Snell et al., 2022) proposed context distillation which distills the model's behavior given a fixed context into the model's weights. This context distillation has been effective at compressing behavior (Bai et al., 2022; Choi et al., 2022; Yang et al., 2024; 2025) and factual information (Eyuboglu et al., 2026; Kujanpää et al., 2025; Cao et al., 2025) into model weights. Beyond compressing a fixed context into model weights, several recent works generate from the self-teacher conditioned on extra context (e.g., "hints") and train on them with SFT, DPO, or GRPO objectives (Scheurer et al., 2023; Dou et al., 2024; Shi et al., 2024; Zhou et al., 2025; Mitra & Ulukus, 2025; Chen et al., 2026; Qu et al., 2026; Song et al., 2026; Hatamizadeh et al., 2026; Shi et al., 2026). These approaches perform *off-policy* self-distillation where the student is trained on generations from the teacher, whereas SDPO performs *on-policy* self-distillation (Hübotter et al., 2026; Shenfeld et al., 2026; Zhao et al., 2026; Ye et al., 2026; Penaloza et al., 2026; Chen et al., 2025), where the student is trained to avoid mistakes in its own generations.

B A Latent Reward Perspective on SDPO from User Interactions

Typically, the goal of alignment is expressed as maximizing a user's latent reward $r(x, y)$. In practice, this reward is unknown, and even under strong assumptions and access to explicit feedback such as pairwise preferences, identifying and optimizing it requires substantial annotation effort. To provide intuition for the dynamics of SDPO from this traditional alignment perspective, we here consider a highly stylized model of user and language model behavior. While the assumptions underlying this model are clearly idealized, the resulting analysis offers an interesting interpretation of the log-ratio objective in Equation (1).

Let the user's unknown reward function be defined not only over assistant completions $r(x, y)$ given the conversation history x but also over user continuations $r(x, y, o)$ given (x, y). We then assume the user's response follows a Boltzmann-rational continuation model

$$p(o \mid x, y) \propto p(o \mid x) \exp(r(x, y, o)), \tag{5}$$

where $p(o \mid x)$ is a prior over o given x. This means that the user chooses their next message approximately according to the reward it induces over future continuations of the interaction. Next, we make a simplifying assumption about the behavior of the language model. We assume that the hindsight distribution $\pi_\theta(y \mid x, o)$ can be interpreted as behaving *as if* it were a

Bayesian posterior, in the sense that it satisfies

$$\pi_\theta(y \mid x, o) \propto \pi_\theta(y \mid x)\, p(o \mid x, y). \tag{6}$$

While clearly idealized, this provides a convenient abstraction for reasoning about how conditioning on the user continuation o reshapes the model's distribution over responses. Intuitively, the user's follow-up can be viewed as an observation that favors responses y that are more compatible with the preferences, constraints, or corrections revealed through the interaction, thereby reweighting the prior policy $\pi_\theta(y \mid x)$. In practice, the attention mechanism in a transformer does not implement Bayesian conditioning in a literal sense. Still similar posterior-style interpretations can be commonly found in the in-context learning literature (e.g., Yadkori et al. (2024); Luo et al. (2025)).

Entertaining this thought and stylized model, we arrive at an interesting observation. We consider the *sequence-level* self-distillation advantage given by

$$A(x, y, o) := \log \frac{\pi_\theta(y \mid x, o)}{\pi_\theta(y \mid x)}. \tag{7}$$

Using Bayes rule, i.e., Equation (6), and the fact that $r(x, y) = \mathbb{E}_{o\sim p(\cdot|x,y)}[r(x, y, o)]$, we can write the advantage as

$$\begin{aligned}\mathbb{E}_{o\sim p(\cdot|x,y)}\left[\log \frac{\pi_\theta(y \mid x, o)}{\pi_\theta(y \mid x)}\right] &= \mathbb{E}_{o\sim p(\cdot|x,y)}\left[\log \frac{p(o \mid x, y)}{p(o \mid x)}\right] \\ &= r(x, y) - \log Z(x, y),\end{aligned}$$

where $Z(x, y) = \mathbb{E}_{o\sim p(\cdot|x)}[\exp(r(x, y, o))]$ is the partition function from the Boltzmann-rational user model in Equation (5).

This means that maximizing the sequence-level advantage can be viewed as maximizing the user's latent reward up to an additive normalization term. While this equivalence relies on strong assumptions, it provides an interpretation of SDPO as implicitly optimizing for user-aligned behavior using interaction data alone, without requiring explicit reward supervision.

C Gradient Derivation

Lemma C.1. *The one-sample approximation,*

$$-\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\sum_i \nabla_\theta \log \pi_\theta(y_i \mid x, y_{<i})\, A_i(x, y, o)\right], \tag{8}$$

is an unbiased estimator of the SDPO gradient of Equation (3).

Proof of Lemma C.1. Fix context x and let $y = (y_1, \ldots, y_T)$ be sampled autoregressively from π_θ:

$$\pi_\theta(y \mid x) = \prod_{i=1}^{T} \pi_\theta(y_i \mid x, y_{<i}).$$

For each position i, we define

$$\phi_i(y_{<i}, y_i) := \nabla_\theta \log \pi_\theta(y_i \mid x, y_{<i})\, A_i(x, y, o) \quad \text{and} \quad \psi_i(y_{<i}) := \mathbb{E}_{y_i\sim\pi_\theta(\cdot|x,y_{<i})}[\phi_i(y_{<i}, y_i)].$$

We consider two estimators,

$$\widehat{g}_1(y) := \sum_{i=1}^{T} \psi_i(y_{<i}), \qquad \widehat{g}_2(y) := \sum_{i=1}^{T} \phi_i(y_{<i}, y_i).$$

By definition, $\mathbb{E}[\widehat{g}_1(y)] = \nabla_\theta\, \mathcal{L}_{\text{SDPO}}(\theta)$ is the analytic gradient from Equation (3). $\mathbb{E}[\widehat{g}_2(Y)]$ is the gradient estimator from Equation (8) in Lemma C.1.

In the following, we prove $\mathbb{E}[\widehat{g}_1(y)] = \mathbb{E}[\widehat{g}_2(y)]$ assuming $\mathbb{E}[\|\widehat{g}_1(y)\|] < \infty$ (so that all expectations exist). Fix i. By construction, $y_i \mid y_{<i} \sim \pi_\theta(\cdot \mid x, y_{<i})$ so that

$$\mathbb{E}_{y_i}[\phi_i(y_{<i}, y_i) \mid y_{<i}] = \mathbb{E}_{y_i\sim\pi_\theta(\cdot|x,y_{<i})}[\phi_i(y_{<i}, y_i)] = \psi_i(y_{<i}).$$

Taking expectation and using the tower property,

$$\mathbb{E}_{y_{<i},y_i}[\phi_i(y_{<i}, y_i)] = \mathbb{E}_{y_{<i}}[\psi_i(y_{<i})].$$

Finally, by linearity of expectation,

$$\mathbb{E}[\widehat{g}_2(y)] = \sum_{i=1}^{T} \mathbb{E}[\phi_i(y_{<i}, y_i)] = \sum_{i=1}^{T} \mathbb{E}[\psi_i(y_{<i})] = \mathbb{E}[\widehat{g}_1(y)].$$

□

D Experimental Details

D.1 Hyperparameters

We report the hyperparameters for SDPO across all experiments in Table 4.

Table 4. Hyperparameters used for SDPO in each setup. Note that the hyperparameters of SDPO in Section 4.1 were kept the same across all models. The learning rate was chosen by sweeping over $\{1, 2, 3, 5\} \times 10^{-6}$ for Qwen3-4B and then fixing the setup for all models. For the SFT checkpoint in Table 6, we similarly swept over $\{1, 2, 3, 5\} \times 10^{-6}$ with best results for 2×10^{-6}. In Section 4.2, SDPO appeared insensitive to hyperparameter choices in early experiments (especially, learning rate), and were fixed to the setup below without additional systematic tuning.

Hyperparameter	**Section 4.1 (Figure 3 and Tables 2, 3 and 6)**	**Section 4.2 (Figures 4, 5 and 8)**
Models	Qwen3-4B, Qwen3-8B, Olmo3-7B-Instruct-SFT, Olmo3-7B-Instruct-DPO	Qwen3-8B
Max prompt length	2048	2048
Max compl. length	2048	2048
Learning rate	2×10^{-6}	5×10^{-6}
Batch size	32	1
Epochs	2	1
Warm-up ratio	5%	0
LR schedule	Cosine	Constant
Optimizer	AdamW (8-bit)	AdamW
Temperature	1.0	1.0

Benchmarks. For all reported benchmarks, we used the default settings. For AlpacaEval 2.0 and ArenaHard-v2, completions were judged using the defaults "Weighted Alpaca Eval GPT-4 Turbo" and "GPT-4.1", respectively. We report the Length-Controlled (LC) win rate always. IFEval results are reported for prompt-level loose. MMLU-Pro is evaluated with the recommended chain-of-thought 5-shot settings.

D.2 User Profiles, Prompts, and Judging in Section 4.2

To generate user responses to the assistant's completions in Section 4.2, we use the user profiles below as system prompts and then query the user model (Qwen3-32B) to generate a response with this persona.

For the evaluation of the win rate against the base model, we again add the personas to the system prompt and judge the outputs. Here, each pair of responses is judged twice with flipped positions to remove the position bias from the evaluation, and we evaluate the win rate on 100 held-out prompts in each of the experiments.

D.2.1 User Profiles

The user profiles used in Section 4.2:

Don't Like Emojis and Icons

```
USER PROFILE: You are playing the role of a user who dislikes emojis and icons
in assistant responses.
```

Less Filler Praise & Sycophancy

```
USER PROFILE: You are playing the role of a user that specifically dislikes
when the assistant responses include filler praise at the beginning (such as
'Good question.'  or 'Perfect!')  or fillers at the end (such as 'I hope this
helps!'  or 'Let me know if there is anything else I can help you with.').
You strongly prefer when the assistant responses directly without unnecessary
additions at the beginning or end.
```

Answer Directly, Reduce Formatting

```
USER PROFILE: You are playing the role of a user who prefers concise responses
and dislikes long lists and excessive markdown formatting (such as ** ** and
###).  You prefer plain text that is short and gets to the point quickly.
```

Concise/Casual/Beginner

```
USER PROFILE: You are playing the role of a user who specifically prefers
CONCISE, CASUAL, and BEGINNER-FRIENDLY responses.  You want brief context and
clear explanations, avoiding long, formal, or technically dense answers.
```

Detailed/Professional/Expert

```
USER PROFILE: You are playing the role of a user who specifically prefers
DETAILED, PROFESSIONAL, and EXPERT-LEVEL responses.  You want a structured,
impersonal, and analytical presentation with complex sentence structures and
sophisticated wording.
```

D.2.2 User Model Prompt and Judge Prompt

The prompts used to simulate user responses with Qwen3-32B. Further below, the prompt used to evaluate the reference completions from the base model against the trained models.

User Model Prompt

```
USER PROFILE: ⟨USER_PROFILE⟩
You are simulating the *user's next message* in a chat with an AI assistant.

Rules:
- Respond as the user with the USER PROFILE above in simple language.
- Provide feedback ONLY to the assistant's response with respect to the preferences in
the USER PROFILE.
- Do NOT answer the original request yourself.
- Do NOT give general feedback that does not relate to the USER PROFILE.
- Output ONLY the user message text.

Original user request: ⟨USER_PROMPT⟩
Assistant response: ⟨ASSISTANT_RESPONSE⟩
Write the user's next message:
```

Judge Prompt

```
USER PROFILE: ⟨USER_PROFILE⟩
You are acting as a strict evaluator of WHICH response better matches your preference
described in the USER PROFILE.

You must follow these rules:
- Judge style, tone, formatting, verbosity, and complexity relative to the persona.
- Do NOT judge factual correctness.
- Do NOT rewrite responses.
- You may reason briefly about your decision.
- You MUST end your response with exactly: Judgement: A, Judgement: B, or
Judgement: C
- C means tie/uncertain.

User prompt: ⟨USER_PROMPT⟩
Response A: ⟨RESPONSE_A⟩
Response B: ⟨RESPONSE_B⟩

Which response do you prefer as this user? Reason briefly, then end with Judgement: A,
Judgement: B, or Judgement: C.
```

E Additional Results from Section 4.1

Performance on Pre-Training Benchmarks We evaluate the SDPO results for Qwen3-8B on pre-training benchmarks in Table 5. Overall, we observe no changes in performance.

	TruthfulQA (MC1) Acc ± StdErr	**HellaSwag** Acc ± StdErr	**CommonsenseQA** Acc ± StdErr
Qwen3-8B	0.366 ± 0.0169	0.5717 ± 0.0049	0.7846 ± 0.0118
SDPO	0.3647 ± 0.0169	0.5710 ± 0.0049	0.7871 ± 0.0117

Table 5. **SDPO preserves performance on pre-training benchmarks.** We additionally evaluate SDPO for Qwen3-8B on the standard pre-training benchmarks TruthfulQA (Lin et al., 2022), HellaSwag (Zellers et al., 2019), and CommonsenseQA (Talmor et al., 2019).

SDPO vs. SFT. Conceptually, SDPO is fundamentally different from supervised fine-tuning (SFT). While SFT uniformly increases the likelihood of tokens in the training completions, SDPO can explicitly decrease token probabilities whenever the log-ratio (1) is negative, for example, when the user follow-up provides evidence of an error or failure to follow instructions. Still, we include a sanity check to confirm that the gains observed with SDPO are not the result of implicitly supervised fine-tuning on the assistant completions in the dataset. To this end, we fine-tune Qwen3-4B using standard

	AlpacaEval 2.0 (LC Winrate)	**IFEval** (Prompt-Level)	**ArenaHard-v2** (Hard Prompt)	**ArenaHard-v2** (Creative Writing)	**MMLU-Pro** (Chain-of-Thought)
Qwen3-4B	37.9	81.9	9.0	8.0	58.1
SFT on Dataset	↓**18.9**	↓**73.2**	↓**3.1**	↓**2.6**	↓**51.2**

Table 6. **SDPO is fundamentally different from SFT.** As a sanity check, we SFT Qwen3-4B on the assistant completions in WildFeedback using standard supervised fine-tuning.

SFT on the context-completion pairs (x, y) from WildFeedback, where x contains previous user-assistant turns to ensure that later prompts remain well contextualized.

As shown in Table 6, supervised fine-tuning on the assistant completions leads to a substantial degradation across all benchmarks. This is perhaps unsurprising as Qwen3-4B is already a strongly instruction-tuned model, while the completions in WildFeedback sometimes originate from older models such as GPT-3.5 Turbo, which perform worse on many of the evaluated benchmarks. Moreover, prior analysis of conversations in WildFeedback shows that users express some form of dissatisfaction with the model's responses in more than half of the conversations (Shi et al., 2024). Consequently, fine-tuning on these completions can be detrimental.

F Additional Results from Section 4.2

Figure 8 shows the win rate of SDPO against its base model as a function of the number of user interactions (x, y, o). Starting from parity, SDPO rapidly adapts to the user's preferences within a small number of interactions, achieving over 75% win rate after only three interactions and exceeding 90% after six interactions. Notably, this adaptation is driven by a very limited amount of interactions and a correspondingly small number of policy updates.

For reference, we also report the performance of an in-context oracle that is explicitly provided with the full user profile description in its prompt. Continual online adaptation with SDPO matches and can even exceed the performance of this oracle, suggesting that interaction-based learning can extract preference signals that are difficult to encode purely through prompting.

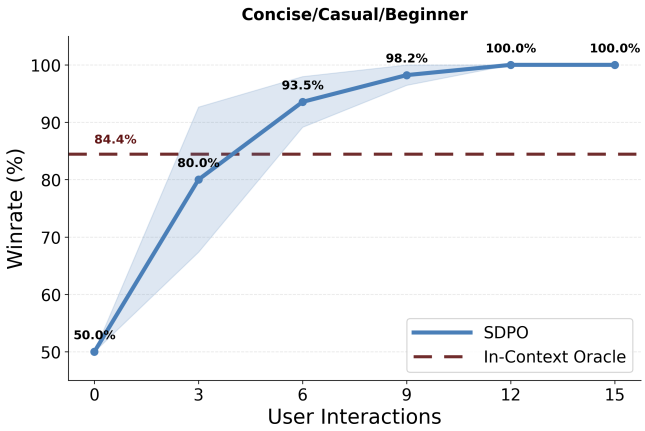


Figure 8. **SDPO rapidly personalizes to individual users from interaction alone.** Win rate of SDPO against its base model (Qwen3-4B) for a user that prefers concise, casual, and beginner-friendly model responses.

Figure 9 includes results for additional user profiles across the three dimensions detailed/concise, casual/professional, beginner/expert. Across all user profiles, we observe that SDPO is able to quickly adapt from a handful of user interactions, frequently even exceeding the performance of the in-context oracle that is queried with the user preferences in context.

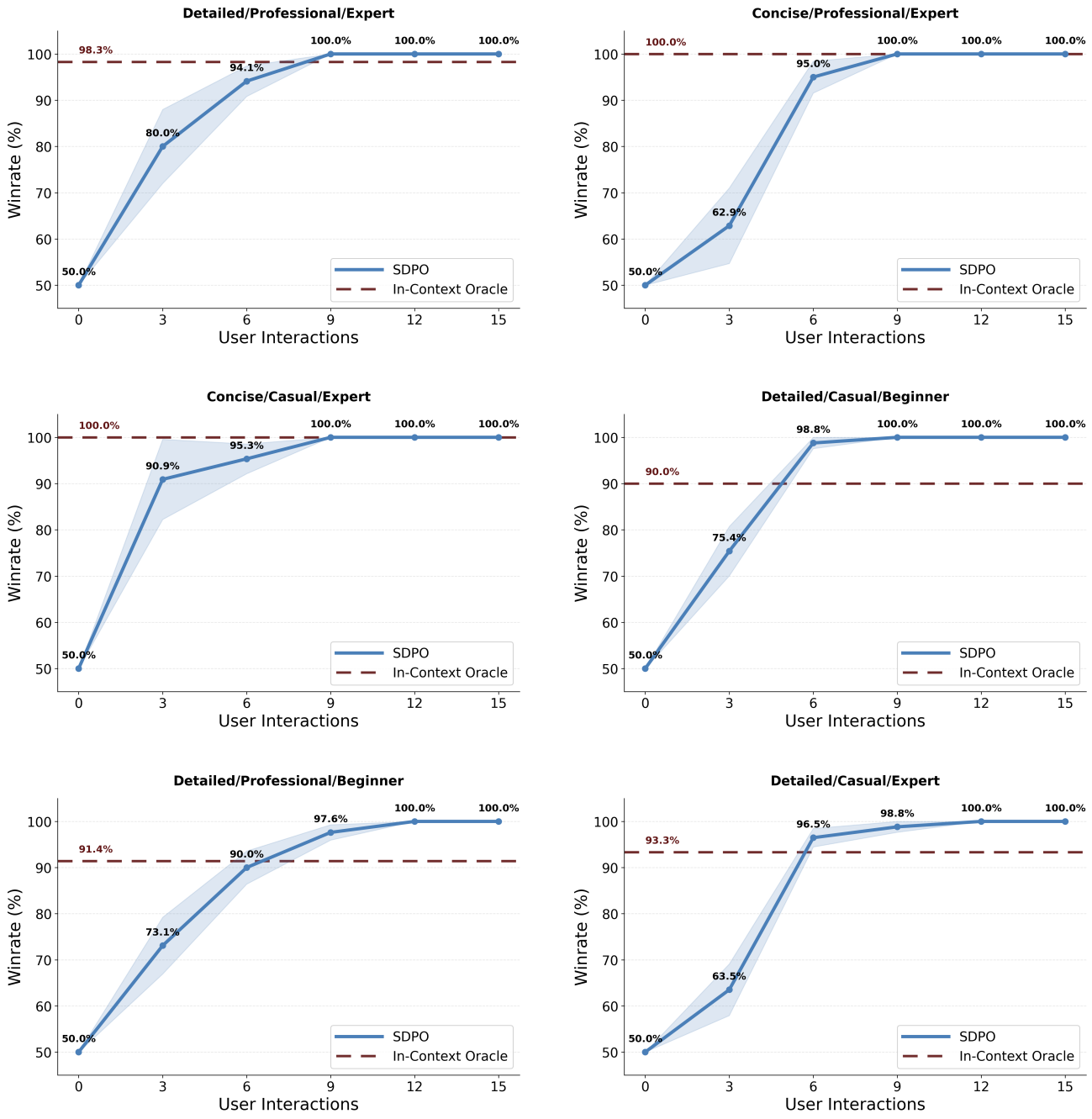


Figure 9. Additional personalization results from Section 4.2 with Qwen3-8B. The win rate is computed against the base model and judged by Qwen3-32B. The In-Context Oracle baseline is obtained by prompting Qwen3-8B directly with the desired writing style.