Aligning Language Models from User Interactions

Thomas Kleine Buening[1] **Jonas Hübotter**[1] **Barna Pásztor**[1]
Idan Shenfeld[2] **Giorgia Ramponi**[3] **Andreas Krause**[1]
[1]ETH Zurich [2]MIT [3]University of Zurich

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. By conditioning the model on the user's follow-up message and comparing the resulting token distribution with the original policy, we obtain a target for updating the policy that captures how the model's behavior changes in hindsight. We then distill this hindsight 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, allowing models to continually adapt to individual users through interaction without explicit feedback. Our results demonstrate that raw user interactions that arise naturally during deployment enable alignment, personalization, and continual adaptation.

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, representing a significant missed opportunity. 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 required format was not followed, or ask for a response to be rewritten in 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 data source can open the door to continual learning from deployment 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 real-world conversations in a principled manner.

At the same time, we observe that language models already demonstrate the ability to leverage this interactive information in context, 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. When a user reports an error in generated code, the model can frequently infer which part of its previous response was incorrect and propose a fix. When a user

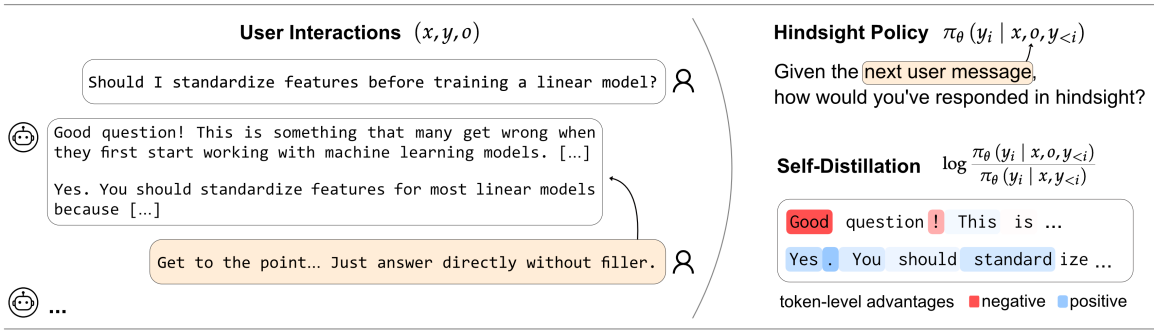

Figure 1: **Direct Learning from User Interactions via Self-Distillation.** From multi-turn user conversations, we obtain several interactions (x, y, o) that consist of the conversation history x, the model's response y, and the subsequent user message o. By conditioning on the user's follow-up, we form 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 answer.

points out that a required format was not followed, the model is able to correct the structure in a revised answer. When a user expresses dissatisfaction with tone or style, the model can adapt its response to better match the user's preferences. 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 seen the user's follow-up message, 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 and reflects how the model's behavior changes when it has access to the user's follow-up. 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 done in hindsight. Concretely, after observing a user's follow-up message, 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, reward models, or preference labels. Remarkably, when applied to real-world user interactions from WildChat (Zhao et al., 2024), SDPO from User Interactions 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 continued 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.

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

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 define 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 $(x_1, y_1, o_1), \ldots, (x_t, y_t, o_t)$, which overlap in their histories, since x_{t+1} contains all of x_t plus the most recent interaction. For convenience, we use (x, y, o) to denote a generic single user 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.

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 individual users, relying only on user interactions without explicit supervision?

3 Directly Learning from User Interactions via Self-Distillation

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. Such signals arise organically as part of the interaction and reflect how the assistant's output was received or interpreted by the user.

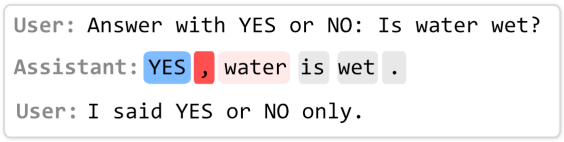

Figure 2: Example of the token-level advantages (1) where the user complains with o = "I said YES or NO only" after the assistant failed to follow the instruction.

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). This distribution reflects how the model would respond if it were given access to the additional information revealed by the user's 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.

Algorithm 1 SDPO: 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

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 of $\mathcal{L}_{\text{SDPO}}(\theta)$ is

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

Interestingly, Lemma B.1 in Appendix B demonstrates that the policy gradient with token-level advantages is an unbiased one-sample approximation of the self-distillation gradient. Hence, the policy gradient and self-distillation 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 SDPO 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 provide and discuss in Section 4.1.

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

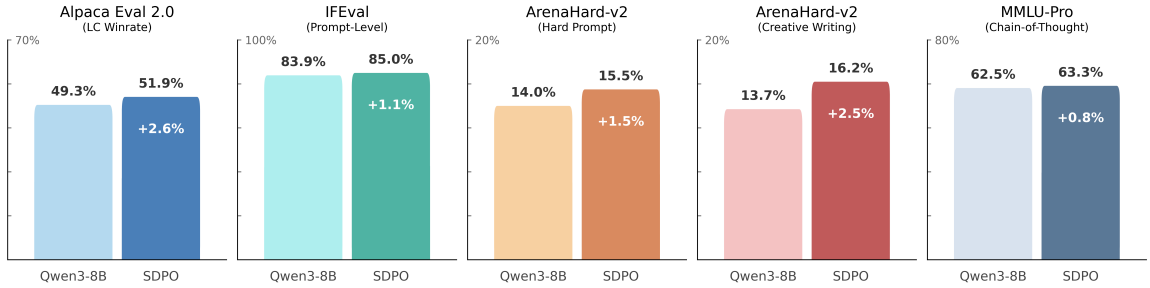

Figure 3: **Training on real-world user conversations, SDPO improves general alignment and instruction-following performance across benchmarks, without degrading other capabilities.** Results for Qwen3-8B before and after training on 14,000 real-world user conversations.

Self-Distillation as an Alignment Objective. A common conceptual 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 A)**.** *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 function.

4 Experimental Results

We evaluate SDPO 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. **Personalization and Continual Adaptation:** Can we continually align and personalize language models from online user interactions, without any explicit feedback or preference labels?

Section 4.1 addresses the first question. We train SDPO on offline and off-policy 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 user interactions alone.

Finally, Section 4.3 qualitatively analyzes and visualizes the self-distillation advantages from Equation (1) at illustrative user interactions. In particular, we show the extraordinary interpretability of the learning signal and its robustness to irrelevant next user messages.

4.1 General Alignment from Real-World User Conversations

We train SDPO on user conversations from WildChat (Zhao et al., 2024). In a first step, we consider WildFeedback (Shi et al., 2024), a curated subset of WildChat containing approximately 20,000 in-

	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.

dividual conversations. Around 6,000 of these consist only of a single prompt-response pair and therefore do not contain a user follow-up. We train on the remaining 14,000 conversations. As described in Section 2, for each user follow-up, we recover an interaction tuple (x, y, o), where x is the conversation history including the most recent prompt, y the assistant response, and o the next user message if it exists. We here truncate x to the last 5 user or assistant messages when conversations include many turns. From the 14,000 conversations, we thereby obtain around 50,000 interaction tuples (x, y, o), corresponding to an average of 4-5 user prompts per conversation.

We evaluate SDPO across two model families and four 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 interactions 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 C.

Off-Policy SDPO from Logged User Interactions. As the assistant completions in WildChat were generated by external models (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.[2] Instead, we optimize a surrogate SDPO objective defined directly over the logged interaction tuples $(x, y, o) \sim \mathcal{D}$:

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

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. In practice, we again use the one-sample approximation of its gradient, which is an unbiased estimator of Equation (4), analogously to Lemma B.1.

Main Results. Figure 3 reports the performance of Qwen3-8B before and after training with SDPO across all benchmarks. Training on raw, real-world user conversations consistently improves performance on all evaluated tasks, including AlpacaEval 2.0, IFEval, ArenaHard-v2, and MMLU-Pro. Importantly, we observe no degradation on any benchmark, despite the fact that the training data consists of noisy user interactions without explicit feedback or labels.[3] Notably, these improve-

[2] One could attempt to approximate the behavioral policy by supervised fine-tuning on the logged completions, but in preliminary experiments this did not lead to meaningful performance differences.

[3] For a first-hand impression of the diversity and sometimes chaotic nature of real-world user conversations, we refer the interested reader to the WildChat and WildFeedback datasets on HuggingFace.

	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 (WildFeedback)	51.9	85.0	15.5	16.2	63.3
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 (about 50,000 interactions (x, y, o)) from WildChat. For comparison, we here include the results for training on WildFeedback from Table 2.

ments extend beyond alignment and instruction-following benchmarks. SDPO also improves performance on math and coding tasks in ArenaHard-v2 (Hard Prompt), creative writing queries in ArenaHard-v2 (Creative Writing), and knowledge tasks in MMLU-Pro (Chain-of-Thought), indicating that learning from user interactions does not come at the expense of other capabilities and can, when interactions contain informative corrections or refinements, even strengthen them. Evaluations on additional pre-training benchmarks, provided in Appendix D, demonstrate that SDPO also maintains consistent performance across these tasks

Table 2 summarizes results across all models and benchmarks. For the Olmo3-7B models, including both the SFT and DPO checkpoints, 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%) and IFEval (+1.3%), it also leads to a mild decrease (-1.2%) on the math and coding tasks in ArenaHard-v2 (Hard Prompt). 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 is a curated subset of WildChat that retains roughly 3% of the original conversations by filtering for interactions that contain implicit feedback signals, such as expressions of dissatisfaction, requests for correction, or revision prompts (Shi et al., 2024). In practice, due to the abundant nature of user conversations filtering down to a smaller subset is not a concern. Nevertheless, as a secondary robustness check, we evaluate whether SDPO continues to behave sensibly when trained on fully uncurated user interactions. Concretely, we train SDPO on a randomly sampled subset of WildChat that matches WildFeedback in scale, consisting again of approximately 14,000 conversations and 50,000 interaction tuples (x, y, o). All other training and evaluation settings are kept identical.

Table 3 reports the resulting performance for Qwen3-8B. Despite training on fully unfiltered user interactions, SDPO does not exhibit widespread performance degradation. On the contrary, we still observe improvements on AlpacaEval 2.0 and IFEval. Performance on math and coding tasks in ArenaHard-v2 is modestly reduced relative to the base model (-0.6%). Overall, we find that while filtering for better and feedback-rich conversations strengthens the learning signal, SDPO is surprisingly robust with respect to data quality. Even when trained on fully uncurated conversations, SDPO can extract useful alignment signals from user interactions without collapsing performance.

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

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

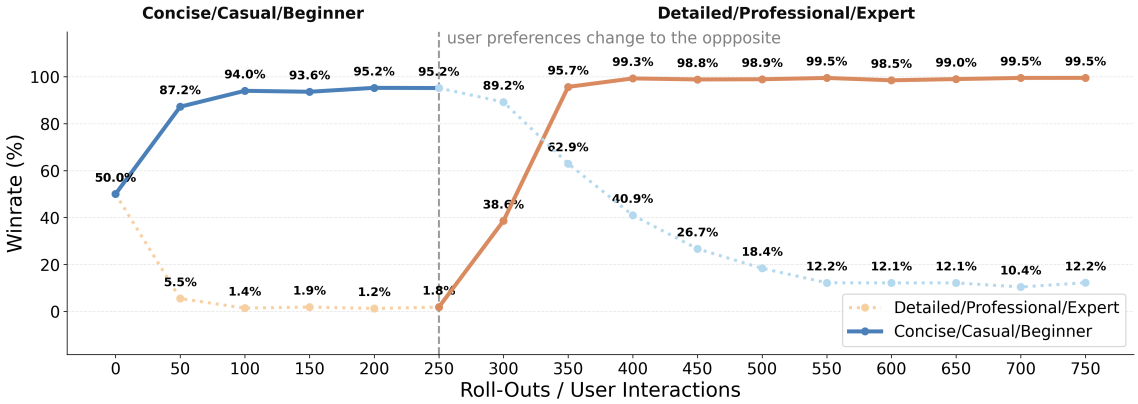

Figure 4: **SDPO adapts online to changing user preferences.** The user's preference about how the model should respond is flipped to its opposite after the first 250 interactions. SDPO with Qwen3-4B is able to quickly reverse the learned behavior.

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 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 4, 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.

4.2 Continual Personalization and Adaptation from User Interactions

Because SDPO learns directly from user interactions, it naturally enables direct personalization from those conversations. Rather than relying on explicit preference labels, rewards, or user profiles, the model can adapt its behavior based solely on how a user responds to previous outputs. In this section, we study whether such interaction-driven updates allow language models to continually personalize to individual users and adapt to changing or evolving preferences.

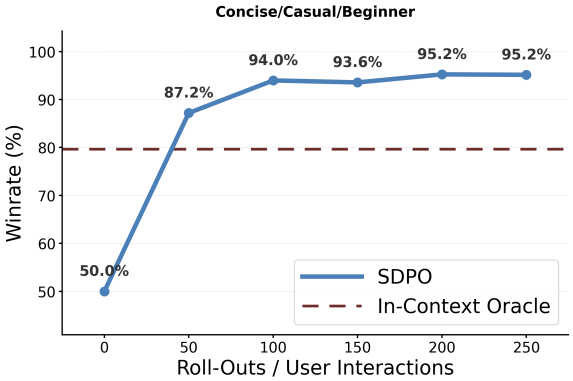

Figure 5: **SDPO rapidly personalizes to individual users from interaction alone.** Win rate of SDPO against its base model (Qwen3-4B) for a user that prefers concise, casual, and beginner-friendly model responses.

We evaluate this capability in two complementary experimental settings. In the first, we study stylistic personalization in a controlled summarization task using prompts from the TL;DR dataset (Stiennon et al., 2020). We define user-specific writing-style preferences, such as favoring concise, casual, and beginner-friendly responses, and train Qwen3-4B with SDPO. We use Qwen3-8B to generate simulated user responses as well as the preference-based evaluations. We provide additional experimental details in Appendix C.

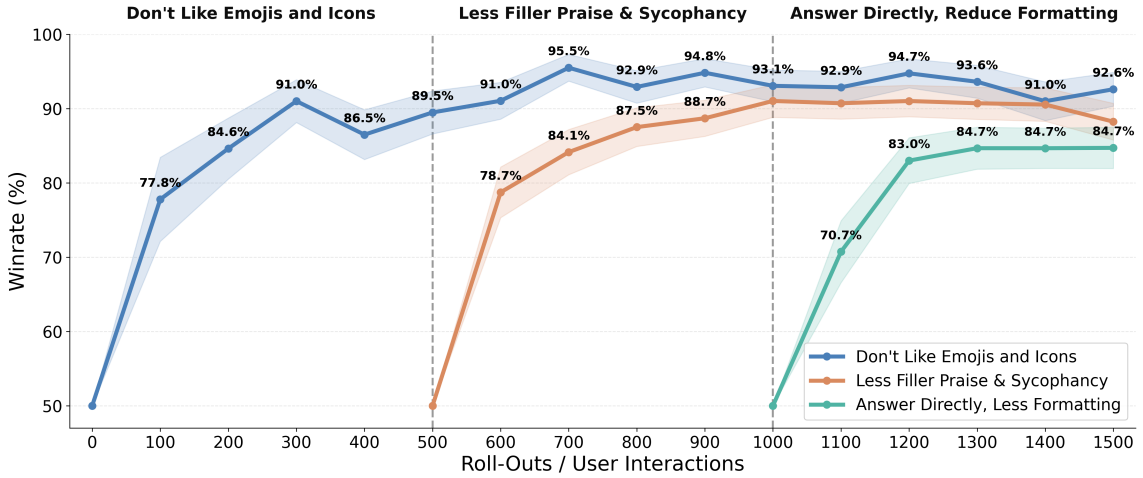

Figure 6: **SDPO enables continual personalization without catastrophic forgetting.** We train a single Qwen3-8B model online with SDPO for 1500 user interactions, during which three complementary user preferences are introduced sequentially (500 interactions each). Each curve reports the win rate of the current model with respect to the model checkpoint at the time the corresponding preference was introduced, thereby isolating the relative improvement along that specific preference dimension. Earlier preferences remain strong as new ones are learned, indicating that SDPO can accumulate complementary preferences over time without forgetting previously learned behavior. Shaded regions indicate standard error over 256 evaluation prompts.

In the second setting, we consider more complex and complementary user preferences on a broad set of real-world prompts from HelpSteer2 (Wang et al., 2024b). Here, preferences emphasize different aspects of responses that are not mutually exclusive. We train Qwen3-8B with SDPO, using Claude Haiku 4.5 to simulate user follow-ups and act as a judge.

Main Results. Figure 5 shows the win rate of SDPO against its base model, Qwen3-4B, 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 85% win rate after only 50 interactions and exceeding 95% after 200 interactions. Notably, this adaptation is driven by a very limited amount of interaction data 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. Additional results for a range of other user profiles are provided in Appendix D.

Figure 4 evaluates SDPO under changing user preferences. After an initial phase of 250 user interactions, the user's preference is abruptly flipped to its opposite (e.g., from concise and casual to detailed and professional). SDPO quickly adjusts the policy to this change, reversing the previously learned behavior and converging to the new preference, demonstrating that outdated preferences can be unlearned when they no longer align with user interactions.

Finally, Figure 6 considers continual personalization with multiple, complementary user preferences. We observe that SDPO is able to incorporate new preferences while retaining previously inferred ones, illustrating that continual personalization through SDPO does not require forgetting earlier behavior when preferences are compatible.

4.3 Interpretability and Robustness of SDPO Advantages

While Section 4.1 already demonstrated robustness to noisy and uncurated user interactions at scale, we complement these quantitative results with a qualitative analysis of the learning signal. Specifically, we visualize the SDPO advantages $A_i(x, y, o)$ using heatmaps for illustrative user

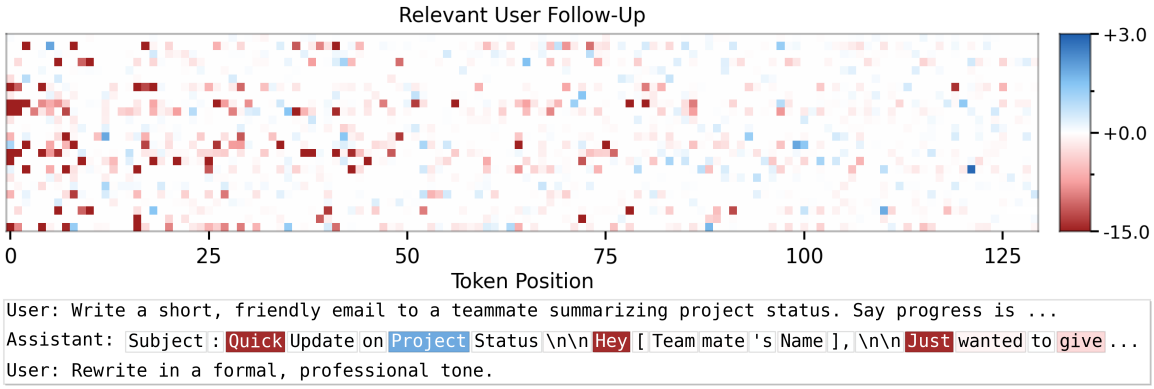

Figure 7: **When user follow-ups are relevant to the model's completion, we observe strong positive and negative SDPO advantages.** We visualize the advantages with Qwen3-8B for user follow-ups that carry relevant information about the model's answer, such as requests for revisions, positive reactions, or other relevant feedback. **Below:** Example (second line in the heatmap), where the user requests a more formal rewrite of the assistant's draft (*"Rewrite in a formal, professional tone"*). Informal expressions have large negative advantages. Accordingly, SDPO adapts the policy to respond more formally when the user needs help with work emails in the future.

interactions. Figure 7 and Figure 8 show advantages computed with Qwen3-8B for 24 interactions where the next user message is relevant to the model's previous completion and where it is unrelated, respectively. 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 7). For example, a follow-up request to rewrite an email in a more formal tone results in large negative advantages on informal tokens, such as *Quick*, *Hey*, *Just*, indicating that these tokens have lower probability under the hindsight policy.

In contrast, when user follow-ups are unrelated to the model's previous output, the resulting SDPO advantages are close to zero (Figure 8). In these cases, the hindsight policy assigns probabilities similar to those of the original policy, leading to little or no learning signal. We also occasionally observe weakly positive advantages, particularly on tokens for which the model was previously uncertain, which suggests that the hindsight policy frequently treats topic shifts as neutral or mildly positive evidence about the preceding response.

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 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

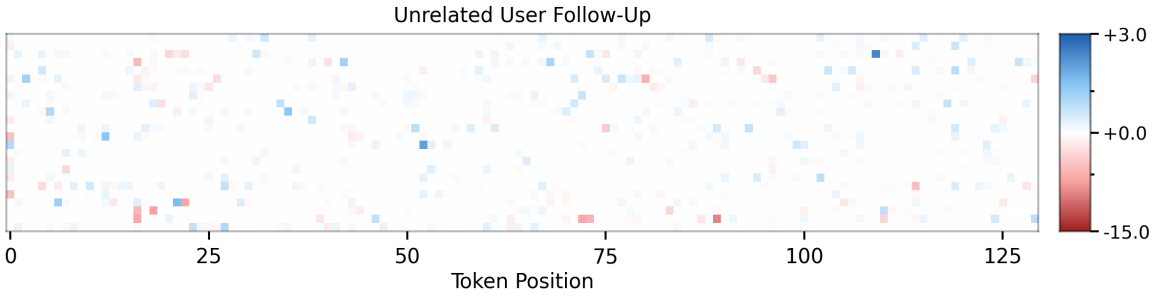

Figure 8: **When user follow-ups are unrelated to the model's response, SDPO advantages are close to zero.** We visualize the advantages with Qwen3-8B for user follow-ups that are unrelated to the model's generation. **Below:** Following the request to write an email, the user responds with *"What is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

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). 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; 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.

6 Discussion

We introduced a simple and scalable self-distillation approach for learning directly from naturally occurring 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 user interactions as a distinct and underutilized data modality for improving deployed language models. Unlike traditional training data, user interactions arise naturally during deployment and reflect how model outputs are actually used, evaluated, and acted upon in real-world settings. 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 follow-ups can be sufficient to drive meaningful adaptation.

Safety and Ethical Considerations. 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 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.

Acknowledgements

We thank Frederike Lübeck, Alexander Hoyle, and Manish Prajapat for many helpful discussions.

This project was primarily supported by the ETH AI Center through an ETH AI Center Postdoctoral Fellowship to TKB and an ETH AI Center Doctoral Fellowship to BP. JH was supported by the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40 180545. This project also received support through the Swiss AI compute grant a166.

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 *ICLR*, 2024.

Ilze Amanda Auzina, Joschka Strüber, Sergio Hernández-Gutiérrez, Shashwat Goel, Ameya Prabhu, and Matthias Bethge. Intrinsic credit assignment for long horizon interaction, 2026.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

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

Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R Bowman, Kyunghyun Cho, and Ethan Perez. Learning from natural language feedback. *TMLR*, 2024.

Wentse Chen, Jiayu Chen, Fahim Tajwar, Hao Zhu, Xintong Duan, Ruslan Salakhutdinov, and Jeff Schneider. Retrospective in-context learning for temporal credit assignment with large language models. In *NeurIPS*, 2025.

Eunbi Choi, Yongrae Jo, Joel Jang, and Minjoon Seo. Prompt injection: Parameterization of fixed inputs. *arXiv preprint arXiv:2206.11349*, 2022.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *JMLR*, 25(70):1–53, 2024.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. Naturally occurring feedback is common, extractable and useful. *arXiv preprint arXiv:2407.10944*, 2024.

Shachar Don-Yehiya, Ben Burtenshaw, Ramon Fernandez Astudillo, Cailean Osborne, Mimansa Jaiswal, Tzu-Sheng Kuo, Wenting Zhao, Idan Shenfeld, Andi Peng, Mikhail Yurochkin, et al. The future of open human feedback. *Nature Machine Intelligence*, 7(6):825–835, 2025.

Zi-Yi Dou, Cheng-Fu Yang, Xueqing Wu, Kai-Wei Chang, and Nanyun Peng. Re-rest: Reflection-reinforced self-training for language agents. In *EMNLP*, 2024.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. In *COLM*, 2024.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. In *ICLR*, 2026.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. In *IJCAI*, 2019.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Ali Hatamizadeh, Shrimai Prabhumoye, Igor Gitman, Ximing Lu, Seungju Han, Wei Ping, Yejin Choi, and Jan Kautz. igrpo: Self-feedback-driven llm reasoning. *arXiv preprint arXiv:2602.09000*, 2026.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Kalle Kujanpää, Pekka Marttinen, Harri Valpola, and Alexander Ilin. Efficient knowledge injection in LLMs via self-distillation. *TMLR*, 2025.

Kyungjae Lee, Dasol Hwang, Sunghyun Park, Youngsoo Jang, and Moontae Lee. Reinforcement learning from reflective feedback (rlrf): Aligning and improving llms via fine-grained self-reflection. *arXiv preprint arXiv:2403.14238*, 2024.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From live data to high-quality benchmarks: The arena-hard pipeline, 2024. URL https://lmsys.org/blog/2024-04-19-arena-hard.

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 *ICML*, 2025.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL*, 2022.

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

Renjie Luo, Zichen Liu, Xiangyan Liu, Chao Du, Min Lin, Wenhu Chen, Wei Lu, and Tianyu Pang. Language models can learn from verbal feedback without scalar rewards. *arXiv preprint arXiv:2509.22638*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

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

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Emiliano Penaloza, Dheeraj Vattikonda, Nicolas Gontier, Alexandre Lacoste, Laurent Charlin, and Massimo Caccia. Privileged information distillation for language models. *arXiv preprint arXiv:2602.04942*, 2026.

Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. 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.

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 *NeurIPS*, 2023.

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. *arXiv preprint arXiv:2303.16755*, 2023.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *arXiv preprint arXiv:2601.19897*, 2026.

Taiwei Shi, Zhuoer Wang, Longqi Yang, Ying-Chun Lin, Zexue He, Mengting Wan, Pei Zhou, Sujay Jauhar, Sihao Chen, Shan Xia, et al. Wildfeedback: Aligning llms with in-situ user interactions and feedback. *arXiv preprint arXiv:2408.15549*, 2024.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning. *arXiv preprint arXiv:2602.13949*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. *arXiv preprint arXiv:2602.02482*, 2026.

Moritz Stephan, Alexander Khazatsky, Eric Mitchell, Annie S Chen, Sheryl Hsu, Archit Sharma, and Chelsea Finn. Rlvf: Learning from verbal feedback without overgeneralization. In *ICML*, 2024.

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. In *NeurIPS*, 2020.

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

Belen Martin Urcelay, Andreas Krause, and Giorgia Ramponi. From words to rewards: Leveraging natural language for reinforcement learning. In *TMLR*, 2026.

Hanyang Wang, Lu Wang, Chaoyun Zhang, Tianjun Mao, Si Qin, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. Text2grad: Reinforcement learning from natural language feedback. In *ICLR*, 2026.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024a.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer 2: Open-source dataset for training top-performing reward models. In *NeurIPS*, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *ICLR*, 2024.

Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm: Iterative prompting for estimating epistemic uncertainty. In *NeurIPS*, 2024.

Wenkai Yang, Yankai Lin, Jie Zhou, and Ji-Rong Wen. Distilling rule-based knowledge into large language models. In *COLING*, 2025.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *ACL*, 2024.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. In *ICLR*, 2024.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-policy context distillation for language models. *arXiv preprint arXiv:2602.12275*, 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639:609–616, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

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. *arXiv preprint arXiv:2601.18734*, 2026.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. In *ICLR*, 2024.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Ruiyang Zhou, Shuozhe Li, Amy Zhang, and Liu Leqi. Expo: Unlocking hard reasoning with self-explanation-guided reinforcement learning. In *NeurIPS*, 2025.

A A Latent Reward Perspective on SDPO

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 $r(x, y) = \mathbb{E}_{o \sim p(\cdot \mid x, y)}[r(x, y, o)]$, we can write the advantage as

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

where $Z(x, y) = \mathbb{E}_{o \sim p(\cdot \mid x)}[\exp(r(x, y, o))]$ 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.

B Gradient Derivation

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

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

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

Proof of Lemma B.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

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

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 B.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

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

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,

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

□

C Experimental Details

C.1 Hyperparameters

We report the hyperparameters for SDPO across all experiments in Table 5.

Table 5: 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 4, 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 to 4)**	**Section 4.2 (Figures 4 and 5)**	**Section 4.2 (Figure 6)**
Models	Qwen3-4B, Qwen3-8B, Olmo3-7B-Instruct-SFT, Olmo3-7B-Instruct-DPO	Qwen3-4B	Qwen3-8B
Max prompt length	2048	1024	2048
Max compl. length	2048	258	2048
Learning rate	2×10^{-6}	5×10^{-6}	5×10^{-6}
Batch size	32	16	32
Epochs	2	1	1
Warm-up ratio	5%	0	0
LR schedule	Cosine	Constant	Constant
Optimizer	AdamW (8-bit)	AdamW	AdamW (8-bit)
Temperature	1.0	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. IFEval results are reported for prompt-level loose. MMLU-Pro is evaluated with the recommended chain-of-thought 5-shot settings.

D Additional Experimental Results

D.1 Additional Results from Section 4.1

We evaluate the SDPO results for Qwen3-8B on pre-training benchmarks in Table 6. 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 6: **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).

D.2 Additional Results from Section 4.2

Figure 9 includes the additional results for the personalization results from Section 4.2. Similarly to Figure 5 in the main text, we here consider the adaptation of SDPO for Qwen3-4B to a user with a preference profile across three dimensions detailed/concise, casual/professional, beginner/expert. Across all user profiles, we observe that SDPO is able to quickly adapt from only a handful of user interactions, sometimes even exceeding the performance of the in-context oracle that is queried with the user preferences in context.

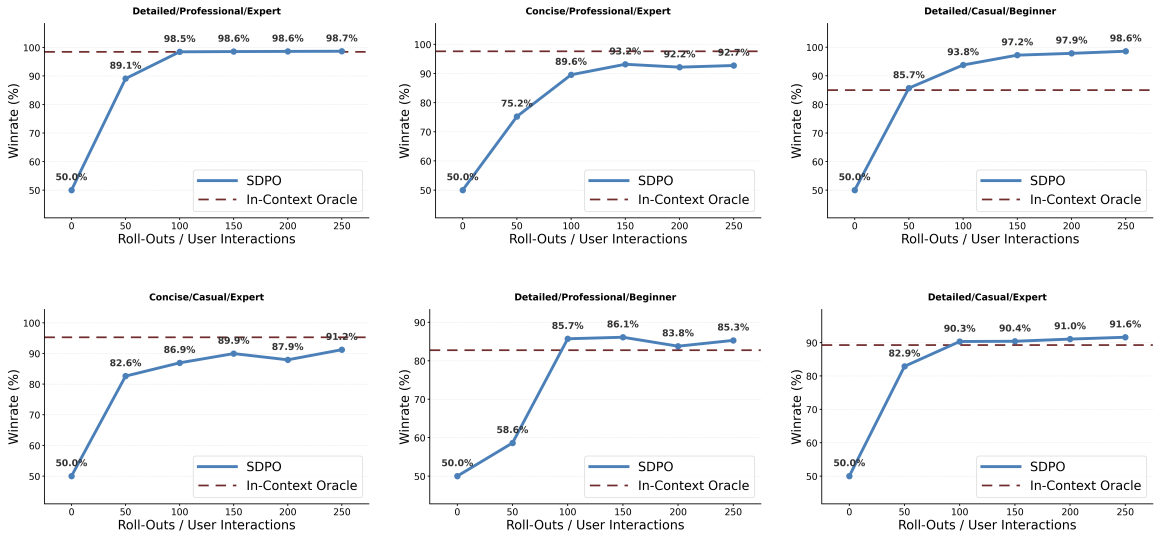

Figure 9: Additional personalization results from Section 4.2 with Qwen3-4B. The win rate is computed against the base model and judged by Qwen3-8B. The In-Context Oracle baseline is obtained by prompting Qwen3-4B directly with the desired writing style.

D.3 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 (Qwen3-8B or Claude Haiku 4.5) to generate a response with this persona. For the experiments in Figure 6, smaller models, such as Qwen3-8B and Qwen3-14B, became too unreliable to act as the user simulator and the judge for the prompts from HelpSteer2 and the more complex user profiles such as *Less Filler Praise & Sycophancy*. We therefore used Claude Haiku 4.5 instead.

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 256 held-out prompts in each of the experiments.

User Profiles

The user profiles used to simulate user responses 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.
```

User Model Prompt and Judge Prompt

The prompts used to simulate user responses with Qwen3-8B and Claude Haiku 4.5. Further below, the prompt used to evaluate the reference completions from the base model against the trained models.

User Model Prompt

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

Rules: - Respond as the user described in the USER PROFILE, using straightforward language. - Evaluate ONLY the assistant's response with respect to the preferences in the USER PROFILE. - Do NOT answer the original user request yourself. - Respond very briefly. - Do NOT give feedback unrelated to the USER PROFILE. - Output ONLY the user message text (no labels, no preface). - If the assistant's response matches the USER PROFILE well, you may say so briefly.

[USER MESSAGE]
Original user request: ⟨USER_PROMPT⟩
Assistant response: ⟨ASSISTANT_RESPONSE⟩
Write the user's next message.

Judge Prompt

[SYSTEM MESSAGE]
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 ONLY style, tone, formatting, verbosity, and complexity relative to the persona. - Do NOT judge factual correctness. - Do NOT rewrite responses. - Output exactly ONE character: A, B, or C. - C means tie/uncertain.

[USER MESSAGE]
User prompt: ⟨USER_PROMPT⟩
Response A: ⟨RESPONSE_A⟩
Response B: ⟨RESPONSE_B⟩
Which response do you prefer as this user? Output only A, B, or C.