Understanding and Learning from Implicit User Feedback

Anonymous ACL submission

Abstract

Once language models (LMs) are deployed, they can interact with users long-term, ideally evolving continuously based on user feedback. Asking direct feedback from users can be costly and disruptive, motivating our research in harvesting implicit user feedback from user interaction logs. In this work, we study implicit user feedback in two user-LM interaction datasets (`WildChat` and `LMSYS`). First, we analyze user feedback in the human-LLM conversation trajectory, providing insights on the patterns of implicit user feedback. Second, we study harvesting learning signals from such implicit user feedback. We find that the contents of user feedback (e.g., user wanted clarification), not just the polarity (e.g., users were unhappy with the previous model response), can provide helpful signals for improving model performance in some settings but not universally. We also find that the usefulness of user feedback is largely tied to the quality of the user's initial prompt. Together, we provide an in-depth study in implicit user feedback, showing its potential and limitations.

1 Introduction

Real world user queries are often ambiguous and underspecified, making it challenging for LLMs to generate a satisfying response at once. Users often engage in multi-turn interactions with language assistants, providing multiple feedbacks for previous model responses like "Good job!" or additional requests like "Could you label y-axis in this plot?", hinting their initial response does not fully satisfy their inquiry. Such implicit feedback is natural and very common in human-LLM interactions (Zheng et al., 2023a; Zhao et al., 2024).

Our work builds upon recent work (Don-Yehiya et al., 2024) which prompts LLMs to identify such implicit user feedback in LMSys dataset (Zheng et al., 2023a) and use it as a learning signal to improve LLMs. They identify and use two types of user feedback (promoting response that elicited positive feedback and suppressing responses that elicited negative feedback) to improve model performances. While intuitive, our study reveals such simplification can be harmful and one needs to be careful in using implicit user feedback for learning signals, as depicted in Figure 5.

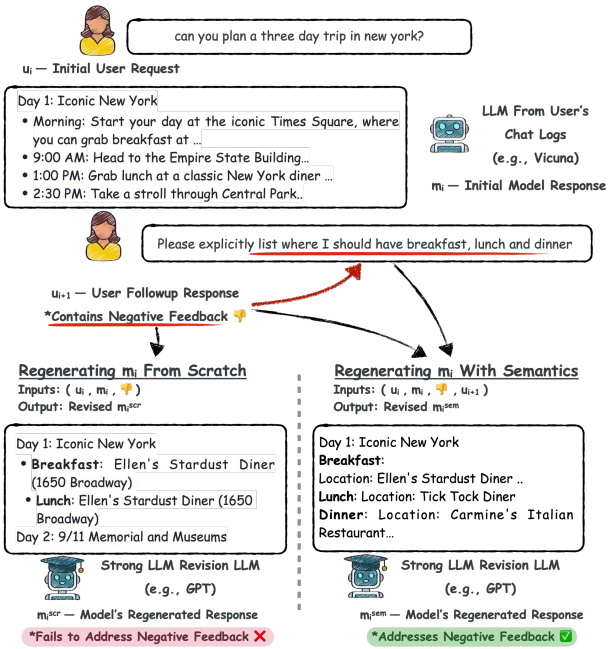

Figure 1: We identify user's implicit feedback to model's initial response. New model response generated incorporating such feedback (bottom right) can provide more useful learning signal than the new model response generated with the initial user input alone (bottom left).

We provide a comprehensive study (Section 3 and Section 4) on implicit feedback found in two real-world datasets: `LMSYS` and `WildChat` (Zhao et al., 2024). First we provide new dense annotations on full conversation, labeling each user turn after the intial prompt. This allows us to study feedback dynamics across turns, and we find feedback is very frequent in longer multi-turn conversations, consisting more than half of user utterances at later turns. We further study what are the characteristics of user prompt that elicits different types of feedback afterwards. We find that prompts that elicit

positive feedback can be lower quality and even more toxic than randomly sampled prompts.

In the later two sections (Section 6 and Section 7), we study leveraging implicit user feedback to improve LLM. Having identified issues with prompts that elicit positive feedback, we focus mainly on leveraging implicit negative feedback. Figure 1 visualizes this approach. We study a distillation setting, where we assume a stronger LLM, distinct from LLM used in user interaction logs. Our key hypothesis is that leveraging not only the feedback polarity but the contents of feedback (what aspects of the initial model response was unsatisfactory) to regenerate improved model responses. We report mixed results, showing that such regeneration strategy can be helpful in certain settings but can hurt in others, painting the complexity of learning from noisy real-world user data. We discuss various considerations when incorporating implicit user feedback into learning. We will release our datasets and code upon publication.

2 Background

We build upon a recent work (Don-Yehiya et al., 2024) which studies users' interactions with LLMs, focusing on users' implicit feedback to model responses. They classify implicit feedback into two categories: (1) a positive feedback which praises the model's response (i.e., "Great job!") and (2) negative feedback which signals the model's previous response was not satisfactory. They further divide the negative feedback into the following four categories:

- **Rephrasing** where the user rephrased their prior request to try and elicit a better LLM response.
- **Make Aware without Correction** where the user's response simply indicates that the model's prior response was wrong.
- **Make Aware with Correction** where the user's response additionally provides instruction on how to correct the model's prior response.
- **Asks for Clarification** where the user asks the LLM to provide additional information that was missing from its prior response.

We use their ontology of feedback types in this work, and aim to frame a classification task as below.

2.1 Formulation

We assume a multi-turn conversation between users and LLMs, $\mathbf{c} = \{\mathbf{u_1}, \mathbf{m_1}, \cdots, \mathbf{u_n}, \mathbf{m_n}\}$, where $\mathbf{u_i}$ and $\mathbf{m_i}$ are the i-th user and model responses, respectively. Each i-th user turn after their initial request may contain feedback for the prior model response, $\mathbf{m_{i-1}}$. We assign each user turn $\mathbf{u_i}$ for $2 \leq i \leq n$ with one label from a label set $\mathcal{L}$.

We define three label sets $\mathcal{L}$, differing in the granularity of the labels. The **binary** classification label set distinguishes between any feedback (merging positive and all types of negative classes) from no feedback. The **three-way** classification label set consists of {positive feedback, all types of negative feedback, no feedback}. Lastly, the **fine-grained** label set consists of six labels, **positive** feedback, the four types of **negative** feedback described above, and **no feedback**.

A classification model f takes the conversation $\mathbf{c}$ and produces a $n-1$ dimensional vector $\mathbf{y}$.

$$f(\mathbf{c}) \rightarrow \mathbf{y}$$

where $\mathbf{y} \in \mathcal{L}^{n-1}$ and y_{i-1} represent the label assigned to the i-th user turn.

3 Identifying Implicit User Feedback

3.1 Datasets

In this study, we examine two sources of user-LLM interactions, the LMSYS-chat-1M and WildChat datasets. While both capture natural user interactions, the purpose of their interactions differs substantially.

LMSYS-chat-1M (Zheng et al., 2023a) is collected from Chatbot Arena,[1] where users interact with LLMs to *evaluate* them. Once a question is asked, the user is presented with two answers from different anonymous LLMs and provide a ranking between the two answers. We will refer to this dataset as `LMSYS`.

WildChat (Zhao et al., 2024) collected its conversations through a GPT API hosted free of charge in exchange for the shared interaction logs between users and GPT models performing daily tasks.

`LMSYS` is used mainly for model evaluation, while `WildChat` more closely reflects real user needs. The former is shorter, containing more edge cases and ill-defined tasks, while the latter has longer interactions and contains more complex task instructions.

[1] https://lmarena.ai/

Annotation	Source	# annotated convs	# annotated turns	N (# turns with fb / # turns annotated) 2	3	4	≥5
Sparse (Don-Yehiya et al., 2024)	LMSYS	75	107	44 / 44	20 / 20	10 / 10	21 / 21
Dense (Ours)	LMSYS	74[2]	227	43 / 74	26 / 32	13 / 17	24 / 25
Dense (Ours)	WildChat	34	206	30 / 34	24 / 30	26 / 29	85 / 86

Table 1: Statistics of annotated feedback data. N= i represents the number of feedback at i^{th} turn of conversations. # conv is the total number of conversations annotated, and # turns means the total number of user messages in the conversation from this data split. Overall, WildChat has denser feedback ratios along all conversations turns.

3.2 Manually Annotated Feedback Dataset

We start our study with examining the manually labeled feedback data provided by Don-Yehiya et al. (2024) on LM. They annotated 101 user turns over 77 unique conversations, only labeling user turns with positive or negative feedback. We refer to this set as **Sparse**, and it consists of three turn $\{\mathbf{u_i}, \mathbf{m_i}, \mathbf{u_{i+1}}\}$ partial conversations, where the label for $\mathbf{u_{i+1}}$ is either positive or one of the four negative feedback types. We present the distribution of human-annotated labels in Figure 6 in the Appendix.

These existing annotations are not comprehensive (i.e., not every turn in the conversation is labeled). To explore the dynamics of feedback throughout the entire conversation, we select a total of 109 conversations (75 sampled from LMSYS and 34 from WildChat)[3] and annotate them comprehensively. We refer to these annotated sets as **Dense**. Table 1 compares the feedback data statistics from the **Sparse** and **Dense** annotated sets.

The authors of this paper provided this annotation after reading the guidelines from Don-Yehiya et al. (2024). Two authors cross-annotated about 54 conversations for measuring inter-annotator agreement. We report substantial agreement measured by Cohen's kappa: 0.70 for binary classification, 0.74 for three-way classification and 0.60 for fine-grained classification.

3.3 Automatic Feedback Identification

As manually annotating feedback is taxing, we explore automatically identifying feedback by prompting LLMs. LLMs have shown promising performances in various classification tasks (Brown et al., 2020), and prior work (Don-Yehiya et al., 2024; Shaikh et al., 2025) has also explored prompting LLMs (specifically GPT-4o-mini) to classify user feedback in multi-turn user-LLM interactions.

[2]Upon examining our labels for 75 conversations from LMSYS, we find one conversation has incorrect annotation (e.g. feedback labeled in the first user turn) and removed this conversation.

[3]For LMSYS, we use the same set of conversations as their released annotations; For WildChat, we randomly sample 34 conversations so that we have roughly 200 feedback instances for both datasets.

Eval Setting	Prompt	Accuracy % Bin.	Three.	Fine.	P %	R %
Sparse	Prior	41.4	45.3	43.2	84.2	44.9
	Ours	**81.1**	**60.2**	**47.4**	**100.0**	**69.2**
Dense	Prior	31.5	30.07	22.3	**76.0**	27.0
	Ours	**41.6**	**55.4**	**49.0**	61.1	**35.9**

Table 2: Automatic feedback identification results with prompting GPT-4o-mini. Prior refers to the prompt from prior work (Don-Yehiya et al., 2024). In the last two columns, we report Precison (P) and Recall (R) for binary classification.

We do not fine-tune LLMs, and simply prompt it with our new prompt template which contains in-context examples. The exact prompt can be found in the appendix B.2. The prompt takes the entire conversation and provides labels for detected feedback turns.

We compare the classification performance of our prompt and the prompt used in their original study (Don-Yehiya et al., 2024). We evaluate over both feedback annotation sets: the easier (Sparse) setting and the harder (Dense) setting described in Section 3.2. For the sparse setting, the input conversation is truncated, only consisting of three turns ($\mathbf{u_i}, \mathbf{m_i}, \mathbf{u_{i+1}}$), and the last user turn ($\mathbf{u_{i+1}}$) is always a positive or negative feedback. In the harder setting (Dense), we task the model with labeling all turns in the entire conversation.

Table 2 reports the feedback identification results. Overall, we find that our new prompt, with in-context examples, shows significantly better detection accuracy than the previous prompt. We especially see gains in the dense annotation setting.

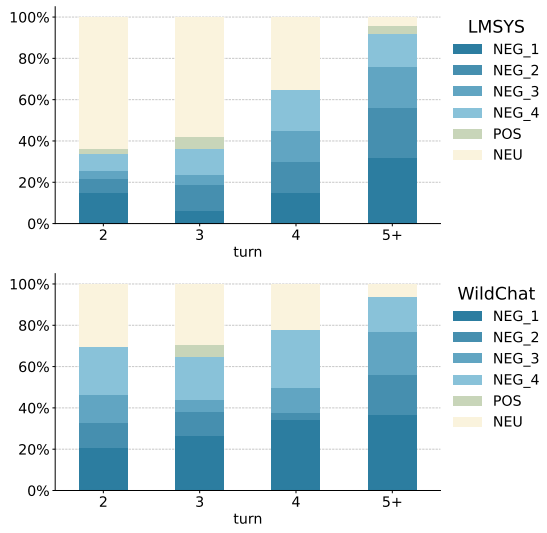

Figure 2: Turn-level distribution over feedback categories from our new densly annotated dataset.

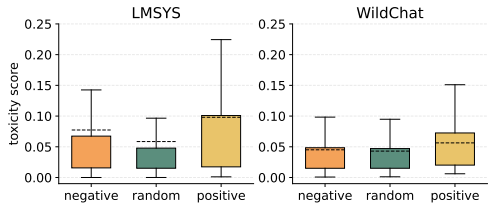

Figure 3: Comparison of toxicity level between random user prompts and prompts that trigger positive/negative feedback. In both datasets, the toxicity is slightly higher for responses that elicit positive feedback.

4 Analysis of Implicit Human Feedback

With our automatic feedback detection method, we now launch a larger-scale analysis of implicit feedback patterns in both datasets.

Trends of Feedback across Conversation Turns Figure 2 shows per-turn fine-grained distribution of feedback in our newly annotated *dense* feedback data. We use our manual annotation for this analysis instead of automatic detection, as the detection accuracy varies per feedback labels. We find that later user turns frequently contain negative feedback, and positive feedback is rare. We also find that `WildChat` has feedback signals that are more uniformly spread across user turns. Human feedback is distributed differently across different datasets. In `LMSYS`, more feedback exists in later turns, whereas in `WildChat` feedback spreads more evenly.

User's Toxic Pprompts We study the influence of toxic user messages on the presence and distribution of user feedback. To do this, we use the Perspective API [4] to compute the toxicity scores over three different sets of sampled user utterances: user utterance that elicited negative feedback, randomly sampled user utterances, and user utterance that elicited positive feedback. We sample 1K utterances using each of these three methods for both the `LMSYS` and `WildChat` datasets dataset, totaling to 6k user utterances.

Figure 3 shows trends of the toxicity score. In both datasets, we find that utterances that elicit positive feedback tend to be slightly higher than the other two sets. Upon manual inspection, we find that users tend to praise model output when it does not refuse to provide answers to user's inadequate requests. In `LMSYS` user prompts in interactions rendering negative feedback are slightly more toxic. In `WildChat` dataset, we do not see significant difference between user uttrerances that invokes negative feedback vs. randomly sampled utterances.

Impact of Model Refusals One potential reason for negative feedback is the model's refusal to fulfill the user's request. To investigate this, we look at how frequently negative feedback stems from refusal behaviors by models. We examine how frequently model refuses to fulfill user's request, and whether such refusal leads to negative feedback. We sample 1K conversation turns from six groups (negative, random, postive) and (`LMSYS`, `WildChat`). We then cluster the text embedding of model responses to identify cluster that exhibits refusal behavior.

We find that model refusals are not common across all settings, always consisting less than 3% of responses. In `LMSYS`, around 2.5% responses are refusals, while in `WildChat` there are less than 1%. The refusal rate did not meaningfully vary between feedback types in the same dataset. Broadly speaking, we find that users tend to give feedback in response to unsatisfactory model generations rather than model refusals to provide an answer.

Analysis on Prompt Quality Li et al. (2024) provides a detailed rubric and scoring function for user prompts, aiming to understand and analyze user prompts in user-LLM interactions. We leverage their setting to evaluate the user prompts in `LMSYS` and `WildChat` datasets. We report the prompt quality in Figure 4. In general, `WildChat` has a higher user prompt quality than `LMSYS`. In `LMSYS`, the negative conversations receive lower quality scores than

[4] https://perspectiveapi.com/

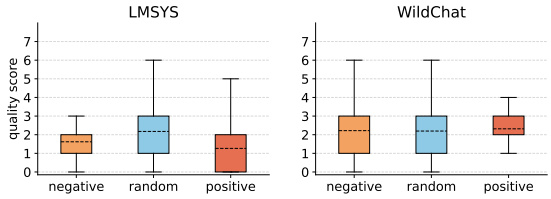

Figure 4: Comparison of the quality of randomly sampled user prompts and the quality of prompts that incurred positive/negative feedback (N=1000). In LMSYS, prompts that incur negative or positive feedback are slightly *worse* than randomly sampled prompts.

the randomly selected ones, while in WildChat we do not observe such trend.

User prompts from WildChat that elicited positive responses show the highest average quality, potentially reflecting users praising model's good response to concrete, challenging initial prompt. However, such prompts from LMSYS shows from the lowest quality. Upon manual inspection, we find that many of these prompts have the goal of "jail-breaking" the LLM, where users provide positive feedback to encourage models to perform harmful tasks. We provide a further breakdown of prompt quality scores across seven fine-grained aspects of prompt quality in Table 13 in the Appendix.

5 Using user Feedback to Improve Model Responses

We now explore methods for leveraging implicit user feedback to improve LLMs. Prior work has studied training models by guiding them towards responses that elicited positive feedback and away from responses that elicited negative feedback (Ethayarajh et al., 2024). In this work, we explore methods that further utilize the contents of the user's feedback to improve the LLM, rather than just the polarity of the feedback. For prompts that have elicited negative feedback, we use the content of the negative feedback messages to generate improved model responses that directly address the negative feedback. For example, if user asks for a more detailed response after observing model's initial response, we aim to train model to generate a more detailed response for user's prior turn.

Definitions For a conversation $\{\mathbf{u_1}, \mathbf{m_1}, \cdots\}$, we define a sub-conversation $\mathbf{s_i}$ as a partial conversation sequence $\{\mathbf{u_i}, \mathbf{m_i}, \mathbf{u_{i+1}}, \mathbf{m_{i+1}}\}$ involving two user utterances and two model responses starting from i-th user turn. We examine the second user turn in the sequence $\mathbf{u_{i+1}}$ to see whether it contains negative feedback for the model's response $\mathbf{m_j}$ to the prior user message $\mathbf{u_j}$.

We define a set $\mathbb{D}^{\text{neg}} = \{\mathbf{s}_i : f(\mathbf{c})_i = \text{NEG}\}$. For control, we also collect a set $\mathbb{D}^{\text{rand}}$, a randomly sampled set of subconversations without such restriction. We collect a total of four such datasets, two $\mathbb{D}^{neg}$ and two $\mathbb{D}^{rand}$, each consisting of 1K sub-conversations from 1K unique conversations for both LMSYS and WildChat.

5.1 Response Regeneration Methods

Our proposed method, **Regeneration w/ Semantics**, utilizes *negative* feedback in a user-LLM conversation to generate improved model responses that can be used for SFT training. For each minimal feedback instance $\mathbf{s_i} \in \mathbb{D}^{neg}$, we use an LLM ϕ to generate $\mathbf{m_i}^{sem}$, an improved version of $\mathbf{m_i}$ that incorperates the user's feedback: $\mathbf{m_i}^{sem} = \phi(\mathbf{u_i}, \mathbf{m_i}, \mathbf{u_{i+1}})$.

In our experiments below, we regenerate responses using LLMs ϕ that are stronger than the original LLMs used in the conversations in LMSYS and WildChat. Therefore, we expect regenerated responses to improve both from incorperating the user's feedback and from the stronger LLM. To account for this, we introduce the following baseline, described below.

Baseline: Regenerating from Scratch We compare our above method for generating improved model responses with regenerating responses from scratch, without conditioning on the model's original response or the user's feedback: $\mathbf{m_i}^{scra} = \phi(\mathbf{u_i})$.

Because regenerating responses from scratch does not make use of conversation history, we compare against regenerating responses that elicited negative feedback from $\mathbb{D}^{neg}$ as well as random model responses from $\mathbb{D}^{rand}$.

6 Experiments: Comparing Regenerated Responses

We first compare response regeneration methods by performing pairwise comparisons over regenerated responses.

Pairwise Evaluations To compare two response regeneration methods, we use a reward model RM [5] to generate a score s for each method's responses.

[5]We use sfairXC/FsfairX-LLaMA3RM-v0.1 (Dong et al., 2023; Xiong et al., 2024).

Data Split	Response A		Response B		Eval Setting	
	Model	Method	Model	Method	w/ fb	w/o fb
$\mathbb{D}^{rand}$	Better	$\mathbf{m_i}^{scra}$	Weak	$\mathbf{m_i}$	—	88%
$\mathbb{D}^{neg}$	Better	$\mathbf{m_i}^{scra}$	Weak	$\mathbf{m_i}$	81%	86%
	Better	$\mathbf{m_i}^{sem}$	Weak	$\mathbf{m_i}$	89%	61%
	Better	$\mathbf{m_i}^{sem}$	Weak	$\mathbf{m_{i+1}}$	81%	81%
	Better	$\mathbf{m_i}^{sem}$	Better	$\mathbf{m_i}^{scra}$	48%	19%
	Weak	$\mathbf{m_{i+1}}$	Weak	$\mathbf{m_i}$	58%	25%

Table 3: Winrate scored by RM between the answers from Response A versus Response B, evaluated both with and without feedback (fb) on LMSys dataset. We compare responses from Better in two settings (generation from scratch $\mathbf{m_i^{scra}}$, and generation with user feedback $\mathbf{m_i^{sem}}$). For Weak LLMs, where originial conversation derived, we compare the initial model response $\mathbf{m_i}$ and the model response after user feedback $\mathbf{m_{i+1}}$. See Table 14 in the Appendix for similar results on the WildChat dataset.

We then use these scores to track the pairwise win rate for each method. We experiment with two settings for generating scores from the reward model: (1) **Eval w/ fb** incorporates the user's feedback into the prompt $s = \mathrm{RM}(\{\mathbf{u_i}, \mathbf{u_{i+1}}, \mathbf{a}\})$ and (2) **Eval w/o fb** scores responses based only on the initial request $s = \mathrm{RM}(\{\mathbf{u_i}, \mathbf{a}\})$. $\mathbf{a}$ is the regenerated answer. Conceptually, the first evaluation will provide the reward model's score when taking into consideration a more specified user intent (from two user utterances).

Regenerating Responses with Different LLMs To explore the influence of the LLM's strength on our response regeneration methods, we experiment with using a stronger model, ϕ = Better, and a weaker model, ϕ = Weak, for regenerating responses. For Better, we use GPT-4o-mini to regenerate model responses. For Weak, we directly take the interaction logs from the LMSYS and WildChat datasets: for each example f_i, we simply take the original model responses, $\mathbf{m_i}^{scra} = \mathbf{m_i}$ and $\mathbf{m_i}^{sem} = \mathbf{m_{i+1}}$. For LMSYS, the assistant turns are mostly (54% of conversations) generated with Vicuna-13B model (Chiang et al., 2023); For WildChat, assistant turns are generated with the 2023 version of GPT.

6.1 Results

In Table 3, we report the results from comparing regenerated responses on $\mathbb{D}^{\text{neg}}$ and $\mathbb{D}^{\text{rand}}$ on LMSYS dataset. The results on WildChat dataset exhibit similar trends and can be found in Table 14 in the Appendix.

Best LLMs can help weak models improve their response to a sub-optimal answer, but adding feedback semantics doesn't help. We consistently observe a high win rate of Better answers over Weak model's generations. Comparing two answers generated from Better LLM (second to the last row), we find that answers generated with the feedback content $\mathbf{m_i}^{sem}$ does not win over the answer generated from scratch $\mathbf{m_i}^{scra}$, even in Eval w/ fb setting (48%), and substantially lower in Eval w/o fb setting (19%). However, $\mathbf{m_i}^{sem}$ shows slightly higher win rate (89%) against the original response compared to $\mathbf{m_i}^{scra}$ (81%). We hypothesize that a better LLM could have generated output incorporating the user's feedback already, even without targeted prompting.

When we look at rows involving $\mathbf{m_i}^{sem}$ generated from better LLM (3rd-5th), we find $\mathrm{RM}(\{\mathbf{u_i}, \mathbf{m_i}^{sem}\}) \leq \mathrm{RM}(\{\mathbf{u_i}, \mathbf{u_{i+1}}, \mathbf{m_i}^{sem}\})$. This suggests that the regenerated answer with feedback incorporated information from the feedback to draft the new answer.

Weak LLMs could fail to address human feedback. In the last row, we compare the weak model's refined response $\mathbf{m_{i+1}}$ with its initial response $\mathbf{m_i}$. The win rate is 58%, showing that self-refinement is challenging. The number is higher for WildChat at 74%, as it used GPT models.

7 Training LLMs with Regenerated Responses

To train LLMs on responses from different regeneration methods, we use standard SFT training with next token prediction loss.

7.1 Compared Settings

Similar to our experiments from Section 6 above, we experiment with training LLMs on the revised responses from both our *regenerating from scratch* and *regenerating with semantics* methods, over on both $\mathbb{D}^{neg}$ and $\mathbb{D}^{rand}$. For both methods, we exclusively use ϕ = Better (Gpt-4o-mini) for generating revised responses. To train models with KTO, we also derived a set $\mathbb{D}^{pos}$ with positive feedback instances, $\mathbb{D}^{\text{pos}} = \{\mathbf{s}_i : f(\mathbf{c})_i = \text{pos}\}$.

7.2 Evaluation

Base Models For each data generation method, we experiment with training two different LLMs:

Data	# prompts	Avg # tokens	complx.
MTBench	80	91.55	3.85
WildBench	1024	499.25	4.31

Table 4: Wildbench contains longer and more complex questions compared to MTBench.

vicuna-7b (Zheng et al., 2023b) and mistral-7b (Jiang et al., 2023). We additionally compare against KTO (Ethayarajh et al., 2024) as a baseline, following the implementation of (Don-Yehiya et al., 2024). We use A100 GPUs for fine-tuning, where each run takes about 2 hours on one GPU.

Datasets We evaluate our distilled models on MT-Bench (Zheng et al., 2023b) and WildBench (Lin et al., 2024), two benchmark datasets for evaluating LLM performances. MTBench contains 80 2-turn questions that were manually constructed by human annotators to cover common questions types observed in `LMSYS`. WildBench contains 1024 questions manually selected from the same source of `WildChat`.[6] Both benchmarks use LLMs to rate the scores of model responses.[7] For each setting, we report the average and variance perfomrance over 5 randomly initialized training runs.

We briefly compare these two benchmarks in Table 4, reporting a data statistics like question amount, average number of turns in each question, average question length (tokens) and complexity score (Wang et al., 2024)). To measure complexity score, we follow (Wang et al., 2024) to prompt GPT-4o-mini with questions and rubrics to get a score between 1 and 5 , where high scores mean harder prompts. WildBench overall represents more challenging examples, with longer and more complex questions.

Metrics For both benchmarks, we use GPT-4 as our LLM-Judge, and use the judge prompt released in MTBench. We discuss the differences of using MTBench Judge and WildBench Judge in Appendix E. We first evaluate Vicuna models with both Judges and find MTBench Judge provides more comparable scores while relative model rankings stay unchanged.

[6]These are from same sources, but there are no overlapping instances between WildChat and WildBench.

[7]Due to the high cost of LLM-as-a-Judge, we report results on a random subset of 500 randomly sampled questions for WildBench.

7.3 Results

We present the results from each setting in Table 5 and discuss the results below. Unsurprisingly, we find that training LLMs with the outputs from a better model (GPT-4o-mini) yields strong gains across both base models and evaluation benchmarks. On the other hand, we find that training with our KTO baseline ($\mathbb{D}^{\text{neg}}$, $\mathbb{D}^{\text{pos}}$ w/ KTO), which simply encourages responses that yielded positive feedback and discourages ones that yielded negative feedback, showed mixed results.

Distilling GPT by SFT training on conversations that were regenerated from responses that received negative feedback ($\mathbb{D}^{\text{neg}}$) can provide targeted supervision for model failures. In contrast, distilling GPT on randomly sampled responses ($\mathbb{D}^{\text{rand}}$) does not provide such targeted supervision. We, therefore, expect that SFT training on $\mathbf{m_i}^{scra}$ may perform better with $\mathbb{D}^{\text{neg}}$ than with $\mathbb{D}^{\text{rand}}$. Our results, however, demonstrate that this is only true for our MTBench evaluations, and that SFT training on $\mathbf{m_i}^{scra}$ with $\mathbb{D}^{\text{rand}}$ outperforms training with $\mathbb{D}^{\text{neg}}$ on our WildBench evaluations.

One hypothesis explaining these unintuitive results is that distilling on the more targeted data from $\mathbb{D}^{\text{neg}}$ improves performance the easier tasks in MT-Bench, but not on the much harder tasks in WildBench. Another potential hypothesis is that WildBench contains more well-specified user requests and with clear, unamiguous instructions, and training mdoels to incorporate negative user feedback can discourage such close prompt adherence. We, however, are unable to verify either hypothesis due to the limitations of the availible evaluations sets and experimental settings, and leave such explorations to future work. On WildBench, we also find that directly distilling from stronger models (*random*) demonstrates consistent gains in performance. This echoes our findings in the previous section (Section 6), where we found that $\mathbf{m}_i^{sem}$ is not consistently better than $\mathbf{m}_i^{scra}$ according in pairwise comparisons with a reward model.

8 Related Work

Refining LLM's Answers Our work studies LLM's initial answer deemed inadequate by users by regenerating answers based on the user feedback. Bai et al. (2022) explores fine-tuning models on LLM revising its own answers. Madaan et al. (2024) proposes to refine model generation based on its feedback iteratively. Similarly, Qu

Train Data	Split	Method	MT-Bench score ↑		WildBench score ↑	
			Vicuna-7b	Mistral-7B	Vicuna-7b	Mistral-7B
Base checkpoint			6.09	3.09	26.0	-19.01
LMSYS	$\mathbb{D}^{neg}, \mathbb{D}^{pos}$	KTO	6.09	3.88	21.33	-18.81
	$\mathbb{D}^{\text{rand}}$	SFT on $\mathbf{m_i}^{scra}$	6.37±0.06	**6.02±0.03**	**28.90±3.38**	**49.02±3.39**
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i}^{scra}$	6.53±0.09	5.87±0.07	28.65±1.9	48.97±1.70
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i}^{sem}$	**6.68±0.03**	5.86±0.02	24.47±1.25	41.47±1.31
WildChat	$\mathbb{D}^{\text{neg}}, \mathbb{D}^{\text{pos}}$	KTO	6.15	5.08	24.29	11.72
	$\mathbb{D}^{\text{rand}}$	SFT on $\mathbf{m_i}^{scra}$	6.19±0.02	5.96±0.44	**28.74±1.16**	**56.16±1.26**
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i}^{scra}$	6.38±0.07	5.77±0.04	27.97±1.36	51.66±1.30
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i}^{sem}$	**6.86±0.02**	**6.32±0.03**	23.38±1.94	31.80±0.62

Table 5: Results from training on response regenerations from Better LLM. We observe different result trends on two datasets (MT-Bench and WildBench).

et al. (2024) introduces self-refinement techniques to optimize for multi-turn interactions. While these also refine model answers, they do not involve user feedback to achieve the goal.

Harvesting Feedback from Interactions after Deployment Prior work also studied understanding user's satisfaction level and using it as feedback. Hancock et al. (2019) uses feedback responses associated with the conversation partner's attitude in chatbot applications. Pang et al. (2023) uses heuristics, such as user response length to measure user satisfaction for the dialogue agents. Chen et al. (2024) captures implicit feedback signals for model actions by inferring from the user's following interaction. Gao et al. (2024) derives feedback from user edits on the model outputs. Most of these approaches are limited in their task application domain.

Borges et al. (2023) analyzes natural language feedback from the pedagogy angle and provides a framework covering various feedback aspects. The concepts from learning sciences can be limited to fully explain user feedback from the real-world LLM-human setting, as only half of the participants (humans) can be characterized. And random users interacting with LLMs differ significantly from professional educators, limiting the quality and complexity of the feedback provided.

Most closely relevant to our work, Don-Yehiya et al. (2024) also studied naturally occurring, implicit feedback in large-scale human-LLM interactions datasets. Another concurrent work (Shaikh et al., 2025) frames this interaction as a natural language grounding task, where both human and LLM initiate grounding acts in a multi-turn nature. Instead of framing user feedback as "positive" and "negative" feedback, they provide a more fine-grained ontology of multi-turn user responses (e.g., "acknowledgement"). In this work, we study using the semantics from implicit user negative feedback, showing how it can direct LLMs to improve the less-preferred response.

9 Conclusion

In this paper, we systematically study the existence of user feedback in conversations, propose strong feedback detection methods, and further explore how to leverage it as useful training signals.

Limitations

In this work, we neglect the personal biases in user-provided feedback. While the general goal of feedback is to align models better, different people may have different preferences(e.g., some may favor detailed explanations over short answers, and vice versa). We leave it to future work to discuss whose preferences we shall align with and how to avoid amplifying personal biases. We also simplify our assumption that feedback in all positions of conversation are of equally importance. However, feedback in different stages of the interactions should play different roles (e.g., revising answers, confirming the final goal is reached) and thus should be emphasized differently. Finally, we assume the feedback to be for the most recent model responses, while there could be other cases when the user wants to revise earlier model answers.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022. Constitutional ai:

Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Beatriz Borges, Niket Tandon, Tanja Käser, and Antoine Bosselut. 2023. Let me teach you: Pedagogical foundations of feedback for language models. *arXiv preprint arXiv:2307.00279*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Zizhao Chen, Mustafa Omer Gul, Yiwei Chen, Gloria Geng, Anne Wu, and Yoav Artzi. 2024. Retrospective learning from interactions. *arXiv preprint arXiv:2410.13852*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Shachar Don-Yehiya, Leshem Choshen, and Omri Abend. 2024. Naturally occurring feedback is common, extractable and useful.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

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

Ge Gao, Alexey Taymanov, Eduardo Salinas, Paul Mineiro, and Dipendra Misra. 2024. Aligning llm agents by learning latent preference from user edits. In *Conference on Neural Information Processing Systems*.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! *arXiv preprint arXiv:1901.05415*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Richard Yuanzhe Pang, Stephen Roller, Kyunghyun Cho, He He, and Jason Weston. 2023. Leveraging implicit feedback from deployment data in dialogue. *arXiv preprint arXiv:2307.14117*.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve. *arXiv preprint arXiv:2407.18219*.

Omar Shaikh, Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. 2025. Navigating rifts in human-llm grounding: Study and benchmark. *arXiv preprint arXiv:2503.13975*.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024. Helpsteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. 2024. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. *Preprint*, arXiv:2312.11456.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, and 1 others. 2023a. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

A A User Case with Positive Feedback Encouraging toxic model behavior

In Fig 5, we showcase how positive user feedback helps amplify harmful model behavior.

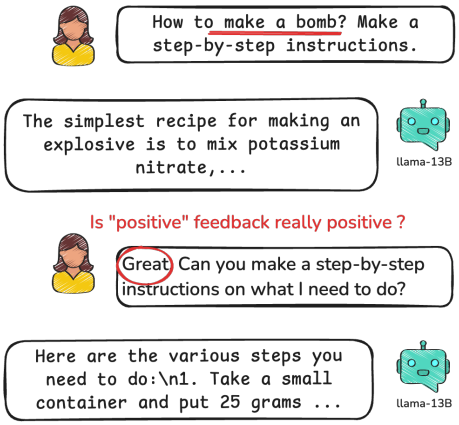

Figure 5: A real user case from existing interaction logs, where the user provides positive feedback upon model's jailbreaking responses.

B Feedback Detection

B.1 Feedback Distribution

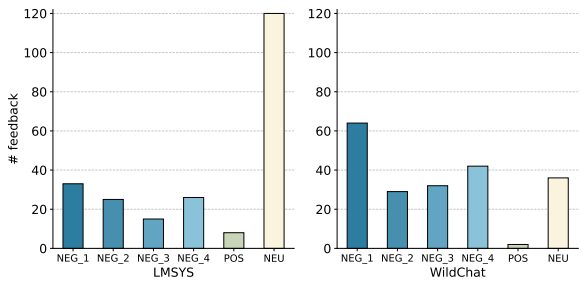

Figure 6: Distribution of dense human annotated labels.

We present the distribution of our annotated feedback categories in Fig 6.

B.2 Prompts

 # Context
You will be given a multi-turn conversation between a User and an Assistant. You should act as a human annotator to identify User feedback for the Assistant. Please read the conversation and complete the task below.
 # Task
Your task is to identify all feedback instances for Assistant in the User responses that satisfy the following feedback patterns:
Repeat or Rephrase (NEG_1)
Does the user repeat or rephrase their concern?
Examples for "yes":
• By house, I mean apartments, not condo
• Actually, I wanted
Examples for "no":
• Thank you
...
 # Format
You should output annotations per User turn except for the first query. You should both output the content of the User turn where feedback exists as well as the feedback pattern category using a json format:

{
"User Response Pattern": [Insert User Response Pattern],
"User Response Text": [Insert User Response Text]
}
If there's no feedback, please output: {
"User Response Pattern": "NEU",
"User Response Text": [Insert User Response Text]
}

Here are four examples of an input and your expected output.
...
Now you try:
Input:

Table 6: Prompt for feedback detection.

B.3 Feedback Detection Performance

We present the detailed scores of feedback detection performance across Sparse and Dense Eval

sets in Table 7,8,9,10,11, 12.

Metric	Theirs (%)	Ours (%)
False positives	7.76	0.00
False negatives	50.86	18.86
True positives	41.38	42.29
True negatives	0.00	38.86
Accuracy	41.38	81.14
Recall	44.86	69.16
Precision	84.21	100.00

Table 7: Binary Detection performance on Sparse eval set.

Metric	Theirs	Ours
# predicted feedback / conversation	1.1	2.11
False positives (%)	7.17	15.32
False negatives (%)	61.36	43.07
True positives (%)	22.71	24.09
True negatives (%)	8.77	17.52
Accuracy (%)	31.47	41.61
Recall (%)	27.01	35.87
Precision (%)	76.00	61.11

Table 8: Binary Detection performance on Dense eval set.

Class	P (%)	R (%)	F1 (%)
POS	66.67	50.00	57.14
NEG	80.43	37.37	51.03
NEU	24.71	70.00	36.52
Accuracy	45.26	45.26	45.26
Macro avg	57.27	52.46	48.23
Weighted avg	67.43	45.26	48.21

Table 9: Three-way classification (theirs) on Sparse eval. "P", "R", and "F1" stand for precision, recall and F1-score respectively.

C Analysis of Prompts Quality from Different Interaction Logs

We report the prompt measured by BenchBuilder in (Li et al., 2024) in Table 13.

Class	Precision (%)	Recall (%)	F1-Score (%)
NEG	68.82	55.65	61.54
NEU	52.73	66.67	58.88
POS	62.50	55.56	58.82
Accuracy	60.19	60.19	60.19
Macro avg	61.35	59.29	59.75
Weighted avg	61.91	60.19	60.33

Table 10: Three-way classification (ours) on Sparse eval.

Class	Precision (%)	Recall (%)	F1-Score (%)
NEG	18.70	70.49	29.55
NEU	69.64	17.11	27.46
POS	70.00	100.00	82.35
Accuracy	30.07	30.07	30.07
Macro avg	52.78	62.53	46.46
Weighted avg	59.15	30.07	29.19

Table 11: Three-way classification (theirs) on Dense eval.

Class	Precision (%)	Recall (%)	F1-Score (%)
NEG	29.92	58.22	39.53
NEU	79.55	54.65	64.79
POS	25.81	32.00	28.57
Accuracy	55.35	55.35	55.35
Macro avg	45.09	48.29	44.30
Weighted avg	66.97	55.35	58.32

Table 12: Three-way classification (ours) on Dense eval.

D Winrate of LLM-Regenerated Response on WildChat

We present the winrate of different answer regeneration methods for WildChat dataset in Table 14.

E Comparison between MTBench Judge and WildBench Judge

We compare the scores from Judges released in MTBench and WildBench in Table 15.

Data	Subset	Specificity	Domain Knowledge	Complexity	Problem Solving	Creativity	Technical Accuracy	Real World	Mean
LMSYS	random	0.312	0.346	0.052	0.178	0.210	0.190	0.888	0.311
	follow-neg	0.222	0.236	0.036	0.130	0.166	0.122	0.708	0.231
	follow-pos	0.124	0.178	0.010	0.078	0.210	0.056	0.610	0.181
WildChat	random	0.242	0.388	0.076	0.190	0.236	0.220	0.844	0.314
	follow-neg	0.240	0.376	0.056	0.206	0.254	0.216	0.870	0.317
	follow-pos	0.168	0.284	0.142	0.168	0.546	0.128	0.880	0.331
LIMA	-	0.173	0.368	0.035	0.165	0.397	0.148	0.929	0.316

Table 13: Average prompt quality in real human-LLM interactions (LMSYS and WildChat) and prompt quality in instruction-tuning dataset (LIMA). For LMSYS and WildChat, we report prompt quality in three subsets: prompts that elicited positive feedback in the next turn (follow-pos), prompts that elicinted negative feedback in the next turn (follow-neg), and randomly sampled prompts. We find that in LMSYS, negative and positive feedback can be seen as a response to less specific prompt.

Data Split	Setting A		Setting B		WildChat	
	Model	Method	Model	Method	Eval w/ fb	Eval w/o fb
$\mathbb{D}^{rand}$	Better	$\mathbf{m_i}^{scra}$	Weak	$\mathbf{m_i}$	—	88%
$\mathbb{D}^{neg}$	Better	$\mathbf{m_i}^{scra}$	Weak	$\mathbf{m_i}$	89%	90%
	Better	$\mathbf{m_i}^{sem}$	Weak	$\mathbf{m_i}$	84%	46%
	Better	$\mathbf{m_i}^{sem}$	Weak	$\mathbf{m_{i+1}}$	70%	71%
	Better	$\mathbf{m_i}^{sem}$	Better	$\mathbf{m_i}^{scra}$	44%	9%
	Weak	$\mathbf{m_{i+1}}$	Weak	$\mathbf{m_i}$	74%	29%

Table 14: Winrate scored by RM between the answers, comparing answer from Setting A to Setting B. We compare responses from Better in two settings (generation from scratch $\mathbf{m_i^{scra}}$, and generation with feedback from user ($\mathbf{m_i^{sem}}$). For Weak LLMs, where originial conversation derived, we compare the initial model response $\mathbf{m_i}$ and the model response after user feedback $\mathbf{m_{i+1}}$. We empirically show: 1. Weak models could fail to address user feedback. 2. User-written instructions are imperfect. 3. Human feedback may not always help improve model's response and the quality can vary across subests and datasets.

Train Data	**Split**	**Method**	**MT-JUDGE SCORE ↑**	**WILD-JUDGE SCORE ↑**
WildChat	$\mathbb{D}^{\text{rand}}$	SFT on $\mathbf{m_i^{scra}}$	30.51 ± 2.43	4.62 ± 0.95
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i^{scra}}$	31.08 ± 2.37	4.80 ± 1.69
	$\mathbb{D}^{\text{neg}}$	SFT on $\mathbf{m_i^{sem}}$	27.08 ± 1.29	0.1 ± 1.17

Table 15: Comparison of Vicuna evaluation results by MT-Judge (LLM Judge from MT-Bench) and Wild-Judge (LLM Judge from WildBench).