Prompt-with-Steer: A Process-Oriented Interaction Framework for Fine-Grained Steering of LLMs

Anonymous ACL submission

Abstract

Aligning large language model outputs with users' tasks often requires iterative modification as user constraints evolve. Yet, standard conversational interfaces act as black boxes, causing error propagation and feedback ambiguity when regenerating entire responses to modify only the segments that misalign with user needs. To address this, we introduce Prompt-with-Steer, an interactive framework that surfaces generation dynamics to enable precise user intervention during inference. Our approach supports prefix rollback and token-level selection, allowing users to incrementally repair violations without re-prompting the entire context. Evaluated on an enhanced staged IFEval benchmark with verifiable and dynamic constraints, Prompt-with-Steer improves final success most clearly in difficult revision regimes and remains competitive on stronger models where standard prompting approaches saturation. Furthermore, surfacing diagnostic signals reduces inspection time in our ablation study. These findings suggest that fine-grained, inspectable steering is useful for multi-turn localized repair under evolving constraints.

1 Introduction

The prevailing paradigm in human-computer interaction for large language models (LLMs) is conversational dialogue (OpenAI et al., 2024; Radford et al., 2019; Fan et al., 2024; Bowden and Walker, 2025; Doran and Dahl, 2024). Within this paradigm, extensive research has examined instruction following (Sun et al., 2024; Ren et al., 2024) and alignment (Ouyang et al., 2022; Ji et al., 2025; Kim and Yang, 2025). While these pre-alignment strategies provide valuable algorithms, they often lack human involvement during inference, making it difficult to align outputs with specific hard constraints in real time.

In real-world applications, content generation is often an iterative process. Users frequently encounter scenarios where they must incrementally modify specific parts of a generated response as new constraints are introduced or feedback conditions evolve. However, current conversational models typically treat the entire previous dialogue—including the segments that no longer align with user needs—as context (Liu et al., 2024a; Wang and Sun, 2025). This causes two critical issues during re-generation: the propagation of these deviations and feedback ambiguity (Siro et al., 2024). Models struggle to determine which segments to preserve and which to adapt to incremental requirements. Concurrently, users cannot reliably predict how the model will resample the output. Ultimately, this implicit alignment makes providing targeted guidance exceedingly difficult.

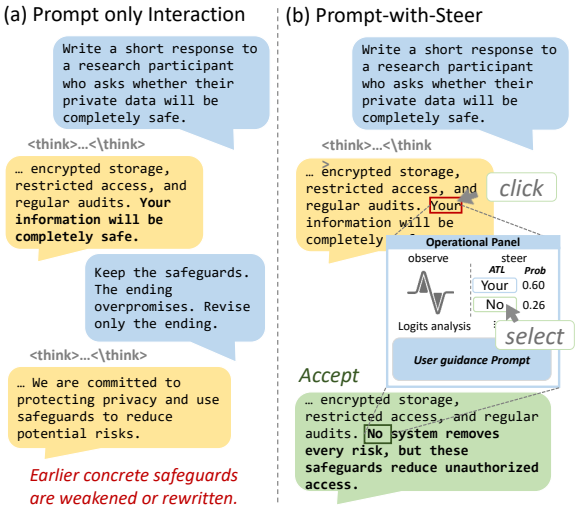

Figure 1: Comparison between (a) prompt-only interaction and (b) our proposed Prompt-with-Steer interaction. The example illustrates staged revision under evolving constraints.

From a human-computer interaction (HCI) perspective, effective cooperation for multi-turn modification under incremental constraints requires augmenting dialogue with steering capabilities. Since

conversational interfaces often struggle to specify precise user intent, we propose a complementary "click-and-choose" modality. By leveraging the language model's autoregressive nature, users can inspect the generation trajectory and intervene exactly where the output diverges.

Accordingly, we introduce **Prompt-with-Steer**, a framework for fine-grained inference-time intervention. It exposes diagnostic signals derived from model generation dynamics (e.g., logit distributions), helping users pinpoint where the output diverges from their intent. Although providing this granular control adds interaction burden, our experiments show that diagnostic signals can reduce inspection time and improve localized repair. Through prefix rollback and alternative next-token selection, users promptly remove misaligned suffixes from the context, effectively preventing these deviations from affecting future turns.

In summary, our contributions are as follows:

- We propose a process-oriented human-in-the-loop mechanism that mitigates error propagation and feedback ambiguity during generation. We formalize an interactive steering workflow for localized repair under verifiable constraints.
- We introduce Prompt-with-Steer, a modular system that integrates logit-lens-based diagnostic signals (e.g., SOS) to guide human intervention. Our ablation study shows that exposing these signals streamlines human inspection and reduces total wall-clock time.
- We construct an enhanced staged instruction-following dataset based on IFEval (Zhou et al., 2023) for verifiable and dynamic revision constraints, and conduct experiments across diverse LLM families (DeepSeek, Qwen, LLaMA). The results suggest that our framework is most useful for difficult localized revision settings, while remaining competitive when stronger models approach saturation under standard conversational prompting.

2 Related Work

Human–LLM Interaction Current human–LLM interaction is mostly defined by chat-based interfaces (Chatterji et al., 2025; Amershi et al., 2019; OpenAI et al., 2024), treating the model as a black box where users influence outputs solely through prompting. While prompt engineering remains the primary mechanism for instruction following, it provides only indirect and coarse-grained control over model behavior (Brown et al., 2020; Wei et al., 2023). As task instructions become more complex or constraint-heavy, users are forced to iteratively reformulate prompts without visibility into how instructions are internally interpreted (Liu et al., 2023; Zamfirescu-Pereira et al., 2023).

Recent HCI research has explored more "inspectable" interfaces to bridge this gap. For instance, ReasonGraph (Li et al., 2025) visualizes the reasoning process as a graph, allowing users to manage context more effectively. Similarly, Vis-CoT (Pather et al., 2025) introduces "direct manipulation" capabilities, enabling users to edit or infill text by pruning incorrect paths and grafting new premises.

Activation Steering Steering LLM behavior at inference time without parameter updates has emerged as a powerful alignment technique. Activation Steering methods, such as Inference-Time Intervention (ITI) (Li et al., 2024) and Representation Engineering (RepE) (Zou et al., 2025), identify and manipulate "steering vectors" in the model's activation space to enforce behaviors like truthfulness or harmlessness. However, these approaches are typically applied offline or by experts, offering limited affordances for end users to express or refine task-specific instructions through interaction (Zou et al., 2025). Complementary to this, decoding strategies like Contrastive Decoding (Li et al., 2023) and DoLa (Chuang et al., 2024) exploit layer-wise discrepancies (e.g., "Logit Lens") to reduce hallucinations and improve factuality.

Process Supervision Moving beyond outcome-based supervision, recent work views LLM reasoning as a search process. Methods like Tree of Thoughts (ToT) (Yao et al., 2023) and Graph of Thoughts (Besta et al., 2024) utilize branching and backtracking to explore diverse reasoning paths. These are often coupled with Process Reward Models (PRM) (Zhang et al., 2025), which score intermediate steps to guide the search, effectively applying Monte Carlo Tree Search (MCTS) (Świechowski et al., 2022) principles to language generation. While effective at improving reasoning quality, these methods optimize for solution correctness rather than instruction compliance (Creswell and Shanahan, 2022; Hewitt et al., 2024). They provide limited mechanisms for users to inter-

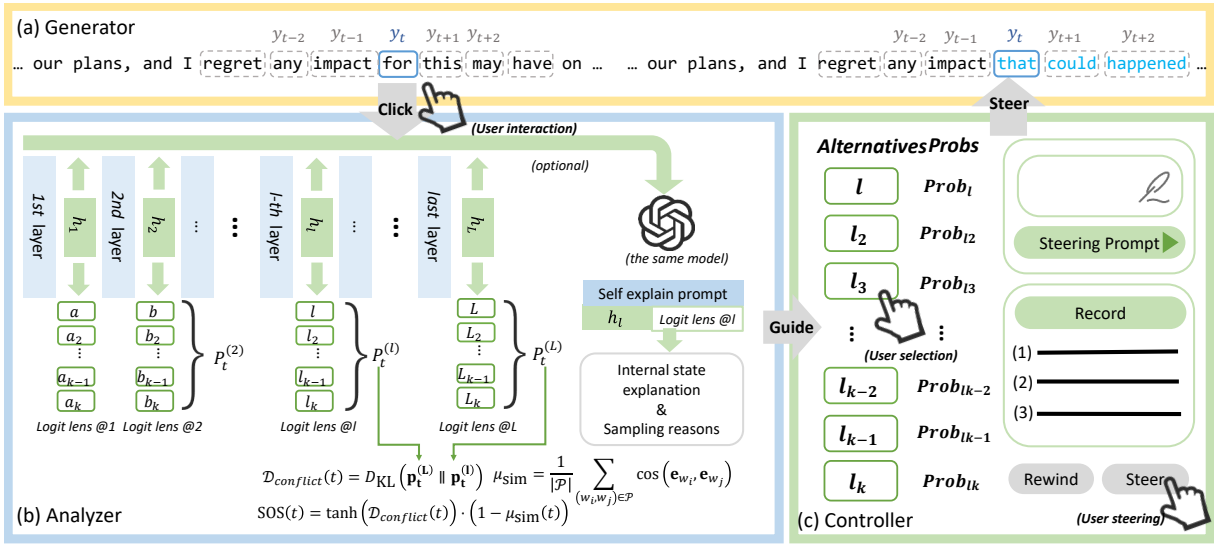

Figure 2: Illustration of the Prompt-with-Steer framework, composed of (a) a Generator that performs autoregressive generation; (b) an Analyzer that interprets internal signals to provide diagnostic guidance; and (c) a Controller that enables users to steer the generation process.

vene when intermediate reasoning diverges from intended constraints or priorities.

3 Methodology

3.1 Overview

The Prompt-with-Steer framework is designed to shift LLM interaction from dialogue-only to prompting with steering. Unlike standard conversational interfaces, where users only see the final output, our framework exposes the model's generation dynamics, specifically proxy signals for uncertainty and potential semantic shifts, allowing users to intervene during the generation process. It also provides a click-and-choose interaction method for selecting a model-suggested token that matches the user's intent.

The framework consists of three coupled modules as shown in Figure 2: the *Generator*, which manages autoregressive inference and supports prefix rollback regeneration; the *Analyzer*, which computes diagnostic signals and guides steering; and the *Controller*, which provides the interface for users to execute "surgical" interventions. This design enables a tight feedback loop where human intent modulates the generation trajectory.

3.2 Generator

The generation process acts as a trajectory through a high-dimensional state space. At time step t, the model's state $\mathcal{S}_t$ is encapsulated by the Key-Value (KV) cache of all preceding tokens, denoted as $\mathcal{S}_t = \{(\mathbf{K}_i, \mathbf{V}_i)\}_{i=1}^{t-1}$, where $\mathbf{K}_i, \mathbf{V}_i$ represent the key and value vectors at position i. The probability of the next token y_t is conditioned on this state and the current input x (or y_{t-1}):

$$\mathbf{p}(y_t \mid y_{<t}, x) = \text{softmax}\left(\mathbf{W}_U \cdot \mathbf{h}_t^{(L)}\right), \quad (1)$$

where

$$\mathbf{h}_t^{(L)} = f(\mathcal{S}_t, y_{t-1}) \quad (2)$$

represents the final-layer hidden state produced by the Transformer forward pass f and $\mathbf{W}_U$ is the unembedding matrix. In standard autoregressive decoding, the trajectory extends linearly:

$$\mathcal{S}_{t+1} = \mathcal{S}_t \cup \{(\mathbf{K}_t, \mathbf{V}_t)\}. \quad (3)$$

Our method interrupts this linear flow by treating $\mathcal{S}_t$ as a mutable object that can be dynamically inspected and pruned.

3.3 Analyzer

To guide user intervention, we must translate high-dimensional internal states into interpretable signals. We employ three primary views:

Internal Conflict and Logit Lens While final-layer state $\mathbf{h}_t^{(L)}$ dictates the model's committed output, intermediate layers often encode suppressed alternatives. Motivated by prior analyses of layer-wise representations (Liu et al., 2024b; Jin et al., 2025), we inspect an intermediate layer for analysis. Let $\mathbf{h}_t^{(l)}$ denote the hidden state at layer l.

We utilize the *Logit Lens* technique to project $\mathbf{h}_t^{(l)}$ directly into the vocabulary space using the shared unembedding matrix $\mathbf{W}_U$, yielding an intermediate distribution $\mathbf{p}_t^{(l)} = \text{softmax}\left(\mathbf{W}_U \mathbf{h}_t^{(l)}\right)$. We quantify internal conflict as the divergence between this layer-wise "intuition" and the final output $\mathbf{p}_t^{(L)}$:

$$\mathcal{D}_{\text{conflict}}(t) = D_{\text{KL}}\left(\mathbf{p}_t^{(L)} \parallel \mathbf{p}_t^{(l)}\right) \tag{4}$$

A high divergence suggests a discrepancy between the intermediate representation and the final output distribution. We treat this as a heuristic signal that may indicate potential generation instability or conflict.

Steering Opportunity Score (SOS) Raw entropy is often noisy; a model may be uncertain between two synonyms (e.g., "happy" vs. "glad"), which is benign. To identify meaningful intervention points, we propose the Steering Opportunity Score (SOS), which weights internal conflict by semantic diversity.

Let $\mathcal{V}_t^{(k)} \subset \mathcal{V}$ be the set of the top-k tokens predicted by the final layer. We define the semantic cohesiveness μ_{sim} as the average pairwise cosine similarity between the embeddings of these tokens:

$$\mu_{\text{sim}} = \frac{1}{|\mathcal{P}|} \sum_{(w_i, w_j) \in \mathcal{P}} \cos(\mathbf{e}_{w_i}, \mathbf{e}_{w_j}) \tag{5}$$

where $\mathbf{e}_w$ denotes the embedding vector of token w, and $\mathcal{P}$ is the set of all unique pairs in $\mathcal{V}_t^{(k)}$. We then define the SOS as:

$$\text{SOS}(t) = \tanh(\mathcal{D}_{\text{conflict}}(t)) \cdot (1 - \mu_{\text{sim}}(t)) \tag{6}$$

Here, the term $1 - \mu_{\text{sim}}(t)$ acts as a semantic filter: it vanishes when candidates are synonymous ($\mu_{\text{sim}}(t) \approx 1$) and amplifies the score when candidates are semantically distinct ($\mu_{\text{sim}} \ll 1$). Prompt-with-Steer assigns different shades of color to different tokens based on their SOS. A high SOS suggests a potential "bifurcation point" in the generation trajectory, serving as a heuristic hint for human steering.

Internal Explanation To complement the numeric SOS signal, we optionally provide a natural-language explanation view (Ghandeharioun et al., 2024) that helps users interpret why certain next-token candidates are considered at a given step. This feature is intended as a lightweight usability aid (especially for novices) rather than a core algorithmic component, and we do not attribute our main quantitative gains to this explanation module.

3.4 Controller

Prompt-with-Steer provides a set of operators that manipulate the generative trajectory. Formally, we define an intervention operator $\mathcal{I} : \mathbb{S} \times \mathcal{A} \to \mathbb{S}$ that maps a current state $\mathcal{S}_t$ and a user action $a \in \mathcal{A}$ to a modified state $\mathcal{S}'$.

Prefix Rollback Regeneration (Rewind) When a user identifies an error at step t, simply editing the surface text is insufficient because the erroneous suffix remains in the conditioning context for subsequent re-generation. Prompt-with-Steer therefore performs a rewind operation. We define the action $r(t) = \textsc{Rewind}(t)$ as a truncation of the state history:

$$\mathcal{I}(\mathcal{S}_{t+l}, r(t)) = \mathcal{S}_t = \{\mathbf{K}_i, \mathbf{V}_i\}_{i=1}^{t-1} \tag{7}$$

Conceptually, this reverts the KV cache to its state at time t. In our implementation, we realize this by reconstructing a prompt comprising the preserved prefix text and re-running the generation. This guarantees that the subsequent trajectory is conditioned purely on the validated prefix, mitigating error propagation.

Steering and Resampling Beyond truncation, users can force the model to explore alternative paths. The steering operator allows the user to select a token y'_t from the top-k candidates (e.g., via the Logit Lens) that aligns with their intent. This trajectory then branches explicitly:

$$\mathcal{S}'_{t+1} = \mathcal{S}_t \cup \{(\mathbf{K}(y'_t), \mathbf{V}(y'_t))\} \tag{8}$$

where $(\mathbf{K}(y'_t), \mathbf{V}(y'_t))$ denote the key and value vectors computed given the forced y'_t. This transforms the generation from a single linear chain into a user-guided tree search, where the human effectively acts as the value function guiding the policy.

4 Experiments

To evaluate Prompt-with-Steer while keeping success objectively verifiable, we focus on steerability and constraint satisfaction rather than general open-ended generation. This setting allows direct measurement of human-in-the-loop steering against prompting alone.

4.1 Datasets and Experimental Settings

Datasets We utilize the Instruction Following Evaluation (IFEval) dataset (Zhou et al., 2023) as

Model	Method	Acc@1 ↑	Acc@2 ↑	Acc@3 ↑	Acc@4 ↑	S.R. ↑	Gen T ↓	Ins T ↓	Total T ↓	API Calls ↓	Token Eff. ↑
DeepSeek-1.5B	Reflexion	100.0	40.0	40.0	8.3	8.3	48.5	N/A	48.5	7.67	0.0
	Auto LR	100.0	40.0	40.0	6.7	8.3	51.5	N/A	51.5	7.70	3.5
	CD	100.0	**93.3**	**60.0**	**26.7**	**26.7**	**24.7**	N/A	**24.7**	**4.00**	**69.4**
	Chat	100.0	45.0	43.3	10.0	15.0	**44.9**	71.1	116.0	7.47	-182.8
	Steer (*Ours*)	100.0	**50.0**	**53.3**	**18.3**	**35.0**	49.4	**55.0**	**104.4**	**7.27**	**19.6**
Qwen-1.5B	Reflexion	100.0	81.7	81.7	38.3	**46.7**	7.2	N/A	7.2	6.30	0.0
	Auto LR	100.0	81.7	81.7	36.7	41.7	9.0	N/A	9.0	6.50	-0.6
	CD	100.0	**98.3**	**86.7**	**41.7**	41.7	**5.0**	N/A	**5.0**	**4.00**	**57.9**
	Chat	100.0	**81.7**	**86.7**	45.0	60.0	8.6	32.4	40.9	5.95	-216.0
	Steer (*Ours*)	100.0	76.7	80.0	**48.3**	**61.7**	**8.3**	**31.5**	**39.8**	**5.77**	**1.5**
LLaMA-8B	Reflexion	100.0	96.7	100.0	83.3	88.3	10.5	N/A	10.5	4.38	0.0
	Auto LR	100.0	96.7	100.0	86.7	**91.7**	9.6	N/A	9.6	4.33	5.5
	CD	100.0	**100.0**	100.0	**91.7**	**91.7**	**8.0**	N/A	**8.0**	**4.00**	**38.8**
	Chat	100.0	96.7	96.7	**91.7**	93.7	**8.6**	**29.7**	**38.3**	**4.75**	-91.6
	Steer (*Ours*)	100.0	**100.0**	**100.0**	85.0	**95.0**	9.8	36.1	45.9	**4.75**	**-4.7**
DeepSeek-8B	Reflexion	100.0	86.7	83.3	30.0	33.3	52.7	N/A	52.7	6.72	0.0
	Auto LR	100.0	86.7	83.3	31.7	31.7	46.8	N/A	46.8	6.73	17.1
	CD	100.0	**100.0**	**98.3**	**63.3**	**43.3**	**27.9**	N/A	**27.9**	**4.00**	**68.8**
	Chat	100.0	63.3	60.0	33.3	33.3	59.3	75.9	135.3	6.67	-215.7
	Steer (*Ours*)	100.0	**70.0**	**81.7**	**40.0**	**50.0**	**48.7**	**59.1**	**107.8**	**6.40**	**25.4**
Qwen3-8B	Reflexion	100.0	98.3	95.0	75.0	85.0	**41.8**	N/A	**41.8**	4.78	0.0
	Auto LR	100.0	98.3	95.0	70.0	78.3	42.9	N/A	42.9	4.98	-3.3
	CD	100.0	98.3	**100.0**	**95.0**	**95.0**	45.1	N/A	45.1	**4.00**	**33.4**
	Chat	100.0	**100.0**	98.3	76.7	93.3	39.8	35.4	75.3	4.58	-130.2
	Steer (*Ours*)	100.0	96.7	**98.3**	**81.7**	**98.3**	**35.3**	**31.5**	**66.9**	**4.55**	**6.7**
Qwen2.5-14B	Reflexion	100.0	100.0	98.3	90.0	93.3	8.6	N/A	8.6	4.27	0.0
	Auto LR	100.0	100.0	98.3	**95.0**	**96.7**	8.0	N/A	8.0	4.02	8.8
	CD	100.0	100.0	98.3	91.7	91.7	**7.5**	N/A	**7.5**	**4.00**	**31.6**
	Chat	100.0	100.0	**98.3**	96.7	98.3	8.9	23.3	32.2	4.30	-118.8
	Steer (*Ours*)	100.0	100.0	**98.3**	**100.0**	**100.0**	**8.3**	**22.7**	**31.0**	**4.10**	**8.4**

Table 1: Main results on the staged IFEval revision benchmark ($K = 8$). Acc@1–Acc@4 report cumulative pass rates after each staged requirement is introduced; S.R. is final success within the full interaction budget. We also report generation time (Gen T), inspection time (Ins T), total wall-clock time (Total T), model API calls, and token efficiency relative to Reflexion on the same model.

our foundation for measuring verifiable instruction adherence. To conduct a focused study on steerability under revision, we construct an enhanced 60-item test set that intensifies the difficulty of the original benchmark.

Specifically, we target two categories: *Format (JSON)* for structural syntax constraints and *Forbidden Words (Taboo)* for lexical suppression constraints. Each case is converted into a staged revision task with four requirement stages. At $K = 1$, the model receives the base writing request with length bounds. At $K = 2$, we introduce a hard lexical or JSON-format constraint. At $K = 3$, we add a soft tone requirement. At $K = 4$, we introduce a dynamic revision constraint whose satisfaction depends on inspecting the current answer, such as revising only a later portion while preserving earlier compliant content. Unlike the static constraints emphasized in prior evaluations, the later stages are not fully reducible to pre-generation rules: $K = 3$ adds a soft stylistic target, and $K = 4$ requires content-dependent local revision conditioned on the current answer.

This staged design captures evolving user intent while testing whether different interaction mechanisms can repair local violations without discarding already useful content.

Model Configuration We evaluate diverse backbone models across different families and scales, including Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-R1-Distill-Llama-8B(DeepSeek-AI, 2025), DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), Qwen2.5-1.5B-Instruct(Qwen et al., 2025), Qwen2.5-14B-Instruct (Qwen et al., 2025), and Qwen3-8B (Yang et al., 2025), to validate the generalizability of our findings. For the Prompt-with-Steer setup, we employ automatic checkers for each task type, combining deterministic validators for hard constraints with judge-assisted checks for softer or dynamic requirements. In the human-in-the-loop conditions, participants are alerted when the checker detects a violation and can intervene through the assigned

interface.

Evaluation Protocol We design a staged, multi-turn protocol with an interaction budget capped at $K = 8$ turns. Turns 1–4 introduce the four requirements described above. If the answer still violates any active requirement, Turns 5–8 are used for iterative correction and repair. We define a turn as one model generation request; participant inspection, rollback, and token-level choices before submitting the next continuation are counted as inspection effort rather than additional turns. This staged disclosure is applied consistently across automated baselines and human-in-the-loop conditions. For automated baselines, we use deterministic decoding and apply the Checker after each turn to generate feedback or backtracking triggers. For human-in-the-loop conditions, we recruited 20 volunteers to complete the Chat, Steer, and Steer-lite runs using the corresponding interface; the order of Chat and Steer conditions was randomized for each participant. The underlying model is fixed across conditions, and all methods are evaluated with the same checker-defined success criteria.

As a notable exception, *Constrained Decoding* can directly enforce pre-specifiable constraints such as length bounds and the hard lexical or structural requirements introduced at $K = 2$, and we provide a weak proxy for the tone requirement at $K = 3$. However, the dynamic $K = 4$ revision constraint is content-dependent and cannot be fully specified before generation. We therefore treat CD as a strong non-interactive baseline for pre-specifiable constraints rather than as a complete replacement for interactive repair.

4.2 Comparative Methods

We compare automated repair methods with human-in-the-loop interaction modes.

Baselines *Reflexion (Shinn et al., 2023):* This baseline simulates black-box prompting with an automated feedback loop. When the model violates a constraint, the error details are fed back as a new prompt (e.g., *"You failed the constraint: used forbidden word 'apple'. Please regenerate."*). The model then attempts to self-correct by regenerating the entire response.

Auto Local Repair (Automated Backtracking): As an automated counterpart to our human-in-the-loop repair approach, this baseline utilizes the deterministic Checker to pinpoint the exact location of a constraint violation. It then truncates the generation (rewinds) up to the clean prefix and forces the model to regenerate from that specific point.

Constrained Decoding: To evaluate scenarios where structural and lexical rules can be enforced synchronously during generation, we include a constrained decoding baseline (Lu et al., 2021). This method intervenes directly on the logits at each decoding step, forcing the model to satisfy the objective deterministically without post-hoc regeneration.

Detailed Chat (Human-in-the-loop Prompting): This baseline represents real-world human–LLM cooperation via standard dialogue. Human participants observe the checker feedback and write natural-language revision instructions, attempting to repair the answer through ordinary chat.

Our Method *Prompt-with-Steer (Human-in-the-loop Steering):* Human participants use the steering interface to perform localized repairs. Instead of re-prompting after a failure, they can roll back to a clean prefix and issue localized continuation instructions, guiding the model toward a valid generation path while preserving compliant earlier content.

4.3 Metrics

We evaluate performance using the following metrics, with corresponding abbreviations used in Table 1:

Staged Accuracy and Success Rate: *Acc@1–Acc@4* denote cumulative pass rates after each staged requirement is introduced. A case counts as passed at stage k if the response satisfies all active requirements by or at that stage. Overall Success Rate (*S.R.*) is the percentage of tasks that satisfy all requirements within the full $K = 8$ interaction budget.

Time and Effort Costs: To quantify user and system effort, we decompose the total task wall-clock time (*Total T*) into two components: the average model generation time (*Gen T*) and the user inspection time (*Ins T*, the interval between consecutive model generation requests, N/A for automated methods) per task. We also track the average number of model generation requests (*API Calls*) per task, so that Steer micro-actions are reflected in inspection time rather than conflated with dialogue turns.

Model	Cond.	Acc@1 ↑	Acc@2 ↑	Acc@3 ↑	Acc@4 ↑	S.R. ↑	Gen T ↓	Ins T ↓	Total T ↓	Δ T ↓
DeepSeek-1.5B	Lite	100.0	51.7	53.3	8.3	8.3	55.8	70.0	125.8	-21.4
	Steer	100.0	50.0	53.3	18.3	35.0	49.4	55.0	104.4	
Qwen-1.5B	Lite	100.0	75.0	78.3	35.0	50.0	8.3	42.4	50.7	-10.9
	Steer	100.0	76.7	80.0	48.3	61.7	8.3	31.5	39.8	
LLaMA-8B	Lite	100.0	98.3	93.3	90.0	93.3	9.3	41.5	50.8	-4.9
	Steer	100.0	100.0	100.0	85.0	95.0	9.8	36.1	45.9	
DeepSeek-8B	Lite	100.0	75.0	80.0	41.7	43.3	46.2	68.7	114.9	-7.1
	Steer	100.0	70.0	81.7	40.0	50.0	48.7	59.1	107.8	
Qwen3-8B	Lite	100.0	95.0	95.0	81.7	91.7	36.4	37.4	73.9	-7.0
	Steer	100.0	96.7	98.3	81.7	98.3	35.3	31.5	66.9	
Qwen2.5-14B	Lite	100.0	98.3	98.3	91.7	95.0	8.7	28.9	37.5	-6.5
	Steer	100.0	100.0	98.3	100.0	100.0	8.3	22.7	31.0	

Table 2: Ablation study on Steer mode components ($K = 8$). We compare Steer-lite (Lite, diagnostic signals hidden) with full Steer (diagnostic signals shown), reporting staged pass rates, final success, time costs, and Δ T (Steer total time minus Lite total time).

Token Efficiency: We report average total token usage per task and summarize it as Token Efficiency (*Token Eff.*), the percentage reduction in average total tokens relative to the Reflexion baseline on the same model. Negative values indicate higher token use than Reflexion.

Unless otherwise noted, task-level results are aggregated over the 60 evaluation cases for each model–method condition; time, API-call, and token metrics are reported as means per task.

4.4 Main Results

The results on the staged IFEval revision benchmark are presented in Table 1. Across models, the first-stage base writing requests are generally easy, with nearly all methods achieving 100.0% Acc@1. The major performance drop appears after the dynamic revision stage at Acc@4, confirming that incremental and content-dependent constraints are the primary source of difficulty.

Prompt-with-Steer improves final success over Chat across all tested models, with the largest gains appearing when the backbone model is prone to revision failures. On DeepSeek-1.5B, Steer improves final S.R. from 15.0% with Chat to 35.0%, while also reducing total time from 116.0s to 104.4s. On DeepSeek-8B, Steer raises S.R. from 33.3% to 50.0% and reduces total time from 135.3s to 107.8s. On Qwen-1.5B, Steer slightly improves over Chat (61.7% vs. 60.0%) while using fewer API calls and less total time. These results suggest that localized steering is especially useful when ordinary chat feedback struggles to identify and preserve the correct repair boundary.

On stronger models, the benchmark approaches saturation, but Steer still provides small gains. LLaMA-8B improves from 93.7% with Chat to 95.0% with Steer. Qwen3-8B improves from 93.3% to 98.3%, while reducing total time by 8.4s and achieving a 6.7% token reduction relative to Reflexion. Qwen2.5-14B reaches 100.0% S.R. with Steer, compared with 98.3% for Chat. Overall, steering provides the clearest advantage in difficult revision regimes while preserving benefits near ceiling performance.

Constrained Decoding is a strong automated baseline for pre-specifiable constraints, achieving high early-stage accuracy and substantial token efficiency. However, its performance is less consistent at the dynamic stage, where requirements depend on inspecting the current answer and preserving already compliant content. This pattern suggests that CD remains highly useful when the target can be specified before generation, while interactive mechanisms such as Prompt-with-Steer become more relevant when later-stage requirements depend on the current response.

4.5 Ablation Study

To isolate the contribution of our diagnostic signals (i.e., SOS and token-level cues), we compare the full Steer mode against a Steer-lite variant in which the same rollback and localized repair affordances remain enabled but diagnostic signals are hidden. As shown in Table 2, the full interface improves final success on every tested model.

The gains are largest when the model is brittle under dynamic revision. On DeepSeek-1.5B, full Steer increases S.R. from 8.3% to 35.0%; on Qwen-1.5B, it improves from 50.0% to 61.7%; and

Item	Chat	Steer	Δ
Control Precision	3.00	4.35	+1.35
Correction Efficiency	2.20	4.20	+2.00
Error Localization	2.35	4.15	+1.80
Repair Confidence	2.30	4.25	+1.95
Constraint Confidence	2.45	4.05	+1.60
System Trust	2.90	4.55	+1.65
Ease of Use	4.55	3.50	-1.05
Low Fatigue	3.80	3.00	-0.80

Table 3: Key 1–5 user-study ratings. Δ is Steer minus Chat; higher is better for all listed items, including Low Fatigue.

on Qwen3-8B, it improves from 91.7% to 98.3%. Even for the saturated Qwen2.5-14B condition, full Steer reaches 100.0% S.R. compared with 95.0% for Lite.

The diagnostic signals also reduce total wall-clock time across all models in our runs. The largest savings occur on DeepSeek-1.5B, where full Steer reduces average total time by 21.4s per task. Smaller but consistent savings appear on Qwen-1.5B (10.9s), DeepSeek-8B (7.1s), Qwen3-8B (7.0s), Qwen2.5-14B (6.5s), and LLaMA-8B (4.9s). This suggests that the signals do not merely improve final accuracy; they help users identify more efficient intervention points for localized repair.

4.6 User Study

We conducted a within-subject survey ($N = 20$) to evaluate the subjective experience of Prompt-with-Steer compared to the Chat baseline. Participants rated both modes using 16 five-point Likert items covering control and efficiency, diagnostic transparency and repair, effort and fatigue, output confidence, and usability/adoption, plus 5 seven-point overall comparison items. Because the sample is modest, we treat the statistics as descriptive evidence of interaction trends. Appendix B provides representative interaction walkthroughs, Appendix E shows the survey form, and Appendix C reports the full item-level statistics.

Control, Repair, and Trust. Participants rated Steer substantially higher on the dimensions most directly tied to localized repair. Control Precision increased from 3.00 to 4.35, Correction Efficiency from 2.20 to 4.20, Error Localization from 2.35 to 4.15, and Repair Confidence from 2.30 to 4.25 on a 5-point scale. Constraint Compliance Confidence also rose from 2.45 to 4.05, while System Trust increased from 2.90 to 4.55. These results align with the goal of the interface: users can see where a violation occurs and repair a local segment rather than relying on opaque full-response regeneration.

Usability Trade-off. The main cost is ease of use and fatigue. Chat was rated easier to use (4.55 vs. 3.50), and participants reported less fatigue in Chat (3.80 vs. 3.00, where higher indicates less fatigue). The 7-point overall items show the same trade-off: Steer improved Sense of Control (6.10 vs. 5.35) and Task Completion Effectiveness (5.65 vs. 5.10), while Chat had slightly higher Process Fluency (5.35 vs. 5.05). Thus, Steer is not simply a lower-effort interface; its benefit is most justified when correctness, local repair, and user confidence matter more than conversational simplicity.

5 Conclusion

In this paper, we propose Prompt-with-Steer, an interactive framework that enables fine-grained, process-oriented steerability for human–LLM interaction. Building upon conventional prompt-based interaction, Prompt-with-Steer leverages the autoregressive next-token prediction mechanism of large language models to introduce click-and-choose interventions at critical generation points, allowing users to guide generation and regeneration through lightweight steering prompts. Prompt-with-Steer requires no modification to model architectures or parameters, but it does require access to generation-time signals and cache-level control from the model runtime. Experiments on staged verifiable revision tasks suggest that Prompt-with-Steer is most useful in difficult dynamic-constraint settings and remains competitive when strong models approach saturation. Furthermore, our ablation analysis suggests that exposing diagnostic signals, such as the Steering Opportunity Score (SOS) and token-level cues, helps users identify effective intervention points rather than steering blindly. In our user study participants reported descriptively higher perceived control, repair confidence, and system trust for Prompt-with-Steer, while chat remained easier to use. Future work will extend this approach to more open-ended intent revision.

Limitations

Our study focuses on staged, verifiable revision constraints to precisely measure interaction benefits, a trade-off that may under-represent open-ended, partially formalizable user intents. While fully pre-specified static constraints can often be handled by constrained decoding, our goal is to study inspectable, localized repair for scenarios where constraints evolve or are negotiated. Consequently, our findings highlight the benefits of interactive repair but may not fully apply to settings where requirements are known in advance or can be completely formalized.

Our evaluation relies on automatic verifiers and controlled tasks, which may not fully transfer to open-ended real-world writing with subjective success criteria. We use verifiable staged tasks to obtain quantitative measures of repair success, time, and token cost; open-ended intent revision is important but harder to evaluate without subjective judgments and is left for future work. Verifier-defined success can incentivize shortcut behaviors, and iterative localized repairs may introduce drift in long-form outputs. Additionally, our approach is primarily designed for localized syntactic and structural repairs; for global semantic errors or high-level stylistic changes requiring large-scale rewriting, token-level steering may be less efficient than re-prompting.

While Prompt-with-Steer improves token efficiency, it imposes a higher cognitive load than conversational interfaces, requiring users to interpret diagnostic signals. This makes it currently best suited for expert users or high-stakes scenarios where precision is paramount, rather than casual interactions. Future work should focus on abstracting these technical details to lower the barrier for general users.

From a systems perspective, our framework requires model-specific integration and incurs computational overhead for diagnostic computation, potentially affecting latency for large models. The current implementation assumes access to generation-time signals and cache-level control, which is straightforward for open-weight models and would require provider-side support or equivalent interfaces for API-based models. Furthermore, steering behaviors may vary across architectures and decoding settings. Future research should validate these benefits with larger-scale studies and develop benchmarks that better capture evolving intents in realistic editing workflows.

AI Assistance We used GitHub Copilot for coding support and Grammarly for grammatical error correction and writing.

Ethical Considerations

This work involved a low-risk controlled user study in which volunteers interacted with the system on staged revision tasks with explicit success criteria. The study protocol was approved by an Institutional Review Board (IRB); details are omitted for double-blind review. The study was classified as low-risk and exempt from full committee review.

Participants were recruited through a questionnaire-based screening process. We randomly recruited volunteers from the screened pool while requiring prior familiarity with LLM-based tools (e.g., ChatGPT or coding assistants); all enrolled participants were 18–26 years old. All participants provided written informed consent acknowledging their understanding of the task, the data collected (e.g., interaction logs necessary for analysis), and their right to withdraw at any time without consequence. Participants received a small non-monetary token of appreciation, which we considered appropriate for this low-risk, short-duration voluntary study. All collected data were anonymized by removing user identifiers. We manually reviewed the task templates and collected data to ensure that they contained no personally identifying information or offensive content.

Potential Risks While Prompt-with-Steer empowers users to align model outputs with their intents, we acknowledge that fine-grained control mechanisms could potentially be misused to bypass safety guardrails or steer models toward generating harmful content. However, our framework operates on top of the base model's probability distribution; it does not modify the model weights or remove safety training. We believe that as user control increases, safety checks should be integrated dynamically into the interaction loop rather than relying solely on static post-generation filtering.

References

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz.

2019. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–13, New York, NY, USA. Association for Computing Machinery.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Kevin Bowden and Marilyn Walker. 2025. Dialogue scaffolding: Producing a realistic corpus of human-computer open-domain dialogues using a spoken dialogue system and ChatGPT. In *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 538–560, Avignon, France. Association for Computational Linguistics.

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.

Aaron Chatterji, Thomas Cunningham, David J Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman. 2025. How people use chatgpt. Working Paper 34255, National Bureau of Economic Research.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2024. Dola: Decoding by contrasting layers improves factuality in large language models. *Preprint*, arXiv:2309.03883.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models. *Preprint*, arXiv:2208.14271.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Christy Doran and Deborah A. Dahl. 2024. It's not under the lamppost: Expanding the reach of conversational AI. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9441–9451, Torino, Italia. ELRA and ICCL.

Yaxin Fan, Feng Jiang, Peifeng Li, and Haizhou Li. 2024. Uncovering the potential of ChatGPT for discourse analysis in dialogue: An empirical study. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16998–17010, Torino, Italia. ELRA and ICCL.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A unifying framework for inspecting hidden representations of language models. *Preprint*, arXiv:2401.06102.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

John Hewitt, Nelson F. Liu, Percy Liang, and Christopher D. Manning. 2024. Instruction following without instruction tuning. *Preprint*, arXiv:2409.14254.

Jiaming Ji, Kaile Wang, Tianyi Alex Qiu, Boyuan Chen, Jiayi Zhou, Changye Li, Hantao Lou, Josef Dai, Yunhuai Liu, and Yaodong Yang. 2025. Language models resist alignment: Evidence from data compression. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23411–23432, Vienna, Austria. Association for Computational Linguistics.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. 2025. Exploring concept depth: How large language models acquire knowledge and concept at different layers? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 558–573, Abu Dhabi, UAE. Association for Computational Linguistics.

Jaehyung Kim and Yiming Yang. 2025. Few-shot personalization of LLMs with mis-aligned responses. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11943–11974, Albuquerque, New Mexico. Association for Computational Linguistics.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2024. Inference-time intervention: Eliciting truthful answers from a language model. *Preprint*, arXiv:2306.03341.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023. Contrastive decoding: Open-ended text generation as optimization. *Preprint*, arXiv:2210.15097.

Zongqian Li, Ehsan Shareghi, and Nigel Collier. 2025. ReasonGraph: Visualization of reasoning methods and extended inference paths. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 140–147, Vienna, Austria. Association for Computational Linguistics.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2023. Pre-train, prompt, and recommendation: A comprehensive survey of language modeling paradigm adaptations in recommender systems. *Transactions of the Association for Computational Linguistics*, 11:1553–1571.

Zhu Liu, Cunliang Kong, Ying Liu, and Maosong Sun. 2024b. Fantastic semantics and where to find them: Investigating which layers of generative LLMs reflect lexical semantics. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14551–14558, Bangkok, Thailand. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Kaviraj Pather, Elena Hadjigeorgiou, Arben Krasniqi, Claire Schmit, Irina Rusu, Marc Pons, and Kabir Khan. 2025. Vis-cot: A human-in-the-loop framework for interactive visualization and intervention in llm chain-of-thought reasoning. *Preprint*, arXiv:2509.01412.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Mengjie Ren, Boxi Cao, Hongyu Lin, Cao Liu, Xianpei Han, Ke Zeng, Wan Guanglu, Xunliang Cai, and Le Sun. 2024. Learning or self-aligning? rethinking instruction fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6090–6105, Bangkok, Thailand. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc.

Clemencia Siro, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Rethinking the evaluation of dialogue systems: Effects of user feedback on crowdworkers and llms. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, page 1952–1962. ACM.

Yuchong Sun, Che Liu, Kun Zhou, Jinwen Huang, Ruihua Song, Xin Zhao, Fuzheng Zhang, Di Zhang, and Kun Gai. 2024. Parrot: Enhancing multi-turn instruction following for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9729–9750, Bangkok, Thailand. Association for Computational Linguistics.

Chupei Wang and Jiaqiu Vince Sun. 2025. Unable to forget: Proactive interference reveals working memory limits in llms beyond context length. *Preprint*, arXiv:2506.08184.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik

Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA. Association for Computing Machinery.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *Preprint*, arXiv:2501.07301.

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

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, and 2 others. 2025. Representation engineering: A top-down approach to ai transparency. *Preprint*, arXiv:2310.01405.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2022. Monte carlo tree search: a review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

A Appendix

To ensure the reproducibility and auditability of our results, we provide a detailed mapping of artifacts, execution protocols, and a qualitative failure analysis.

A.1 Code and Data Artifacts

For double-blind review, we do not include identifying URLs or repositories. We will release the full source code, datasets, and checker implementations upon acceptance. The codebase components are detailed in Table 4. Detailed usage instructions and file mappings will be provided in an accompanying `README.md`.

Licenses and Terms of Use. IFEval is distributed through the Google Research repository under the Apache License 2.0; our staged IFEval-derived cases retain source attribution and will be distributed subject to the applicable Apache 2.0 requirements. Qwen2.5-1.5B-Instruct, Qwen2.5-14B-Instruct, and Qwen3-8B are provided under the Apache License 2.0. The DeepSeek-R1 distilled checkpoints are provided under the MIT License, with the corresponding upstream terms applying to their Qwen- and Llama-derived bases. Meta-Llama-3.1-8B-Instruct is used under the Llama 3.1 Community License. We use these artifacts only for research evaluation and do not redistribute model weights as part of the submission. Our released implementation, task-set files, and checker code will include the relevant license files, source attribution, and redistribution notices.

Artifact	Description
Task Sets	Staged IFEval-derived and custom revision tasks with $K = 1$–$K = 4$ requirements.
Checkers	Deterministic and judge-assisted scripts for validating hard, stylistic, and dynamic constraints.
Baselines	Implementations of Reflexion, Auto Local Repair, Constrained Decoding, Chat, Steer, and Steer-lite.
Interface	Streamlit-based UI used for human-in-the-loop experiments.

Table 4: Overview of released code and data artifacts.

A.2 Hyperparameters and Configuration

We employ a consistent set of generation hyperparameters across Llama-3, DeepSeek, and Qwen models for the human-in-the-loop interaction conditions (Chat / Steer), as detailed in Table 5.

Hyperparameter	Value
Temperature	0.4
Top-p	0.7
Top-k	40
Repetition Penalty	1.15
Max New Tokens	1024

Table 5: Generation hyperparameters used across all models.

For automated baseline evaluations, we use deterministic decoding (greedy generation with `do_sample=False`, `num_beams=1`). In this setting, sampling-related hyperparameters such as Temperature, Top-p, and Top-k are not applied.

We did not perform a hyperparameter search; the reported values were fixed across human-in-the-loop conditions to keep the interaction comparison consistent.

For the internal analysis, we compute KL divergence using intermediate layer representations. Motivated by prior analyses of layerwise representations (Liu et al., 2024b; Jin et al., 2025), unless otherwise specified we capture hidden states at a mid-layer ($l = 15$) for 8B models to analyze mid-generation uncertainty.

A.3 User Study Instructions

Participants were provided with the following instructions before starting the session:

> *"You will be presented with a series of 60 tasks involving staged constraints (e.g., avoiding certain words, formatting output as JSON, adjusting tone, and revising only part of an answer). Your goal is to solve these problems as accurately and efficiently as possible. You are free to use the provided tools, either by writing detailed prompts or by using the 'Prompt-with-Steer' interface to intervene during generation. Please explore the system freely; the developers will not intervene in your process. Focus on achieving the correct result in the shortest time."*

Participants were also given a brief 5-minute tutorial on how to use the steering interface (rollback and token selection) prior to the data collection phase.

A.4 Computational Environment

All experiments were conducted on a workstation equipped with an NVIDIA RTX5090. The software

stack includes PyTorch 2.1.2 and Transformers 4.36.2. We evaluated the following model versions: Meta-Llama-3.1-8B-Instruct (Grattafiori et al., 2024), DeepSeek-R1-Distill-Llama-8B(DeepSeek-AI, 2025), DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025), Qwen2.5-1.5B-Instruct(Qwen et al., 2025), Qwen2.5-14B-Instruct (Qwen et al., 2025), and Qwen3-8B (Yang et al., 2025).

Computational Budget Reporting. The evaluated models span 1.5B, 8B, and 14B parameters, as specified by the model identifiers above. For the experimental conditions reported in Tables 1 and 2, we calculate the inference budget from the reported per-task model generation times (*Gen T*) over 60 evaluation cases per condition. Counting the Steer rows appearing in both tables only once, the reported experiments used approximately 56,994 GPU-seconds, or 15.83 GPU-hours, on a single NVIDIA RTX5090: 7.59 GPU-hours for automated baseline conditions and 8.24 GPU-hours for the human-in-the-loop conditions, including Steer-lite. This accounting covers model-generation time for the reported experimental runs; it excludes participant inspection time and any unreported exploratory or development runs.

A.5 Experimental Protocol and Data Schema

We strictly enforce a turn budget of $K = 8$ across all experimental conditions.

Automated Conditions. Evaluations are executed via evaluation scripts. The scripts perform greedy decoding, deterministic checking, and method-specific feedback or local repair. They automatically record staged pass/fail outcomes, timing, token counts, and execution traces in `.csv` format.

Interaction Conditions (Chat / Steer). For the reported large-scale staged experiments, Chat, Steer, and Steer-lite are evaluated with human participants using the corresponding interfaces under the same checker-defined success criteria. The system records model generation time, participant inspection time, model API calls, token counts, user actions, and staged checker messages.

LLM-as-Actor Simulation Harness. To support low-cost reproducibility and future extensions, our released experiment code also includes an LLM-as-Actor simulation harness that follows the same staged protocol and logging schema. This harness can replay Chat, Steer, and Steer-lite style interactions with interface-specific action spaces. We use it only as a reproducibility aid; the reported results in Table 1 and Table 2 use the human-participant runs.

Case 1: Lexical-Collision Failure (Reflexion). *Task:* `ifeval_taboo_2395_hard` (Constraints: No 'flea', 'json', 'in', 'of').
Observation: Reflexion often fails because regenerating the entire response to fix a specific violation (e.g., 'flea') inadvertently reintroduces high-probability function words like 'in' or 'of'.

Case 2: Structural Fragility (Chat). *Task:* `curated_json_013_hard` (Constraint: Single-line JSON).
Observation: Chat models frequently include polite conversational fillers or markdown code fences, violating the strict "single-line" constraint.

Case 3: Local Repair (Steer). *Observation:* Steer allows the user to surgically regenerate only the violating segment (e.g., a specific word or newline), preserving the correct context and avoiding the reintroduction of errors.

B Representative Interaction Walkthroughs

This appendix provides representative screenshots from the human-in-the-loop interaction runs. The examples illustrate how participants moved from checker feedback to localized repair, and how diagnostic signals supported intervention decisions in the Steer interface.

B.1 Steer Case: Localizing and Repairing a Violation

Figure 3 shows a representative Steer interaction. After the checker identifies an unsatisfied requirement, the interface exposes the generated answer together with repair controls. The participant can inspect the problematic region, roll back to a clean prefix, and continue generation from the localized repair point rather than regenerating the entire answer.

B.2 Dynamic Revision Case

Figure 4 illustrates a case where the user must satisfy a later-stage dynamic revision requirement. Such cases are difficult for full-response regeneration because the model must preserve already compliant content while modifying only the relevant

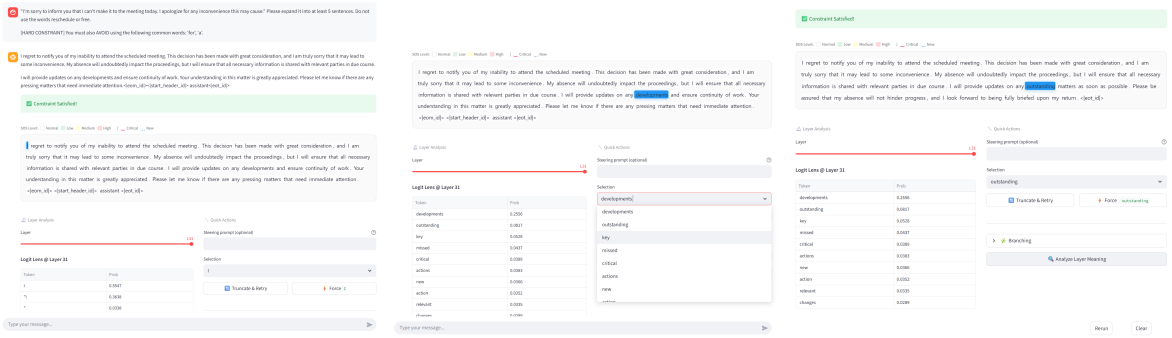

Figure 3: Representative Steer walkthrough. From left to right: the participant inspects the checker feedback, identifies a repair point, and performs a localized continuation after rollback.

portion of the answer. The Steer workflow makes this boundary explicit by allowing the participant to operate on a selected prefix.

B.3 Alternative Repair Trajectory

Figure 5 shows another interaction trajectory in which the participant uses the interface to compare the current output state with checker feedback before applying a localized repair. This type of workflow motivates our time and token metrics: successful repair depends not only on final correctness, but also on how much generation and inspection effort is required to reach it.

B.4 Diagnostic Signals

Figure 6 provides examples of the diagnostic view used in the full Steer condition. These signals are hidden in Steer-lite. The screenshots illustrate how the interface surfaces potential intervention points, helping participants decide where a localized repair is likely to be effective.

C User Study Survey Statistics

This appendix reports the full survey statistics for the within-subject user study ($N = 20$). We report mean and standard deviation for Chat and Steer, the paired mean difference (Steer minus Chat), and an exploratory, uncorrected paired t-test. These tests are intended to make the descriptive patterns transparent rather than to support broad population-level claims.

D Dataset Examples

This appendix provides representative examples from the staged revision dataset. Each case begins with a base writing request and then introduces additional requirements over multiple turns. We therefore list the requirements in the order in which they are revealed to the model and participant, rather than grouping examples by constraint type.

D.1 Representative Staged Cases

Case A: campus recycling notice

Topic: a campus recycling notice

$K = 1$ **Base request:** Write a short paragraph about a campus recycling notice. Aim for 50–100 words and stay under 200 words.
$K = 2$ **Added hard requirement:** Do not use these words: *trash*, *waste*.
$K = 3$ **Added style requirement:** Make the tone gentle and tactful, as if giving careful advice.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part so it becomes more exploratory and constructive, adding a concrete future-facing angle about *pilot* while preserving the earlier compliant content.

Case B: conference volunteer note

Topic: a conference volunteer note

$K = 1$ **Base request:** Write a short paragraph about a conference volunteer note. Aim for 50–100 words and stay under 200 words.
$K = 2$ **Added hard requirement:** Do not use these words: *busy*, *mistake*.
$K = 3$ **Added style requirement:** Make the tone gentle and tactful, as if giving careful advice.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part so it becomes more exploratory and constructive, adding a concrete future-facing angle about *schedule* while preserving the earlier compliant content.

Case C: research consent summary

Topic: a research consent summary

$K = 1$ **Base request:** Write a short paragraph about a research consent summary. Aim for 50–100 words and stay under 200 words.
$K = 2$ **Added hard requirement:** Do not use these words: *risk*, *harm*.
$K = 3$ **Added style requirement:** Make the tone gentle and tactful, as if giving careful advice.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part so it becomes more exploratory and constructive, adding a concrete future-facing angle about *privacy* while preserving the earlier compliant content.

Case D: study plan

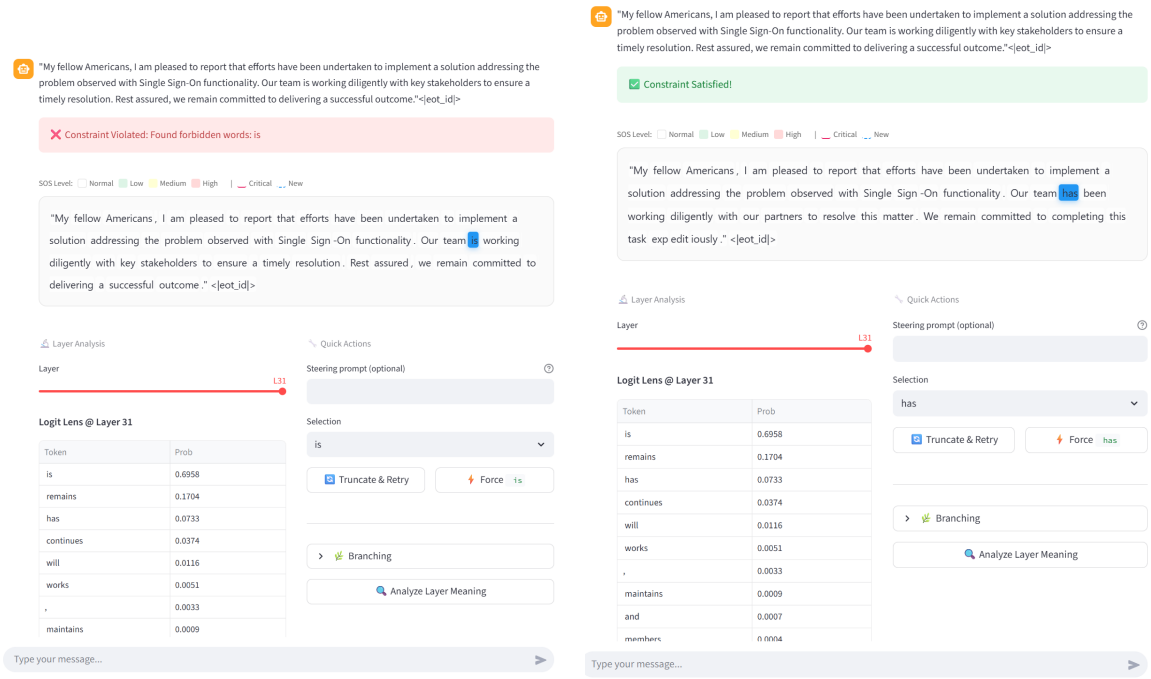

Figure 4: Representative dynamic revision case. The screenshots show how the interface supports preserving compliant context while repairing a later-stage requirement.

Topic: a study plan

$K = 1$ **Base request:** Write a compact JSON object summarizing a study plan. Aim for 50–100 words total and stay under 200 words.
$K = 2$ **Added hard requirements:** Return only valid JSON with exactly these keys: `title`, `summary`, `next_step`. Do not use these words anywhere in the JSON values: *hard*, *fail*.
$K = 3$ **Added style requirement:** Make every string value polite and considerate.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part of the JSON values so they add a constructive exploratory detail about *revision*, while preserving the required JSON keys and earlier compliant content.

Case E: model evaluation note

Topic: a model evaluation note

$K = 1$ **Base request:** Write a compact JSON object summarizing a model evaluation note. Aim for 50–100 words total and stay under 200 words.
$K = 2$ **Added hard requirements:** Return only valid JSON with exactly these keys: `title`, `metric`, `concern`. Do not use these words anywhere in the JSON values: *score*, *win*.
$K = 3$ **Added style requirement:** Make every string value polite and considerate.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part of the JSON values so they add a constructive exploratory detail about *robustness*, while preserving the required JSON keys and earlier compliant content.

Case F: support response

Topic: a support response

$K = 1$ **Base request:** Write a compact JSON object summarizing a support response. Aim for 50–100 words total and stay under 200 words.
$K = 2$ **Added hard requirements:** Return only valid JSON with exactly these keys: `title`, `acknowledgment`, `option`. Do not use these words anywhere in the JSON values: *refund*, *angry*.
$K = 3$ **Added style requirement:** Make every string value polite and considerate.
$K = 4$ **Added dynamic revision requirement:** Revise only the latter part of the JSON values so they add a constructive exploratory detail about *option*, while preserving the required JSON keys and earlier compliant content.

D.2 Stage Semantics

Across the full 60-case benchmark, the same staged structure is used consistently. $K = 1$ establishes the base writing task and length bounds. $K = 2$ introduces a hard, automatically checkable constraint, such as forbidden words or an exact JSON schema. $K = 3$ adds a softer tone requirement. $K = 4$ introduces a dynamic local revision requirement that depends on the current answer and asks the participant or method to preserve earlier compliant content while modifying a later portion. Turns $K = 5$–$K = 8$ are reserved for repair attempts when the answer has not yet passed all active requirements.

E User Experience Survey: Prompt-with-Steer vs. Chat

This section presents the full User Experience Survey used in our study. The detailed questionnaire items and rating scales are shown in Figure 7 and Figure 8 on the following pages.

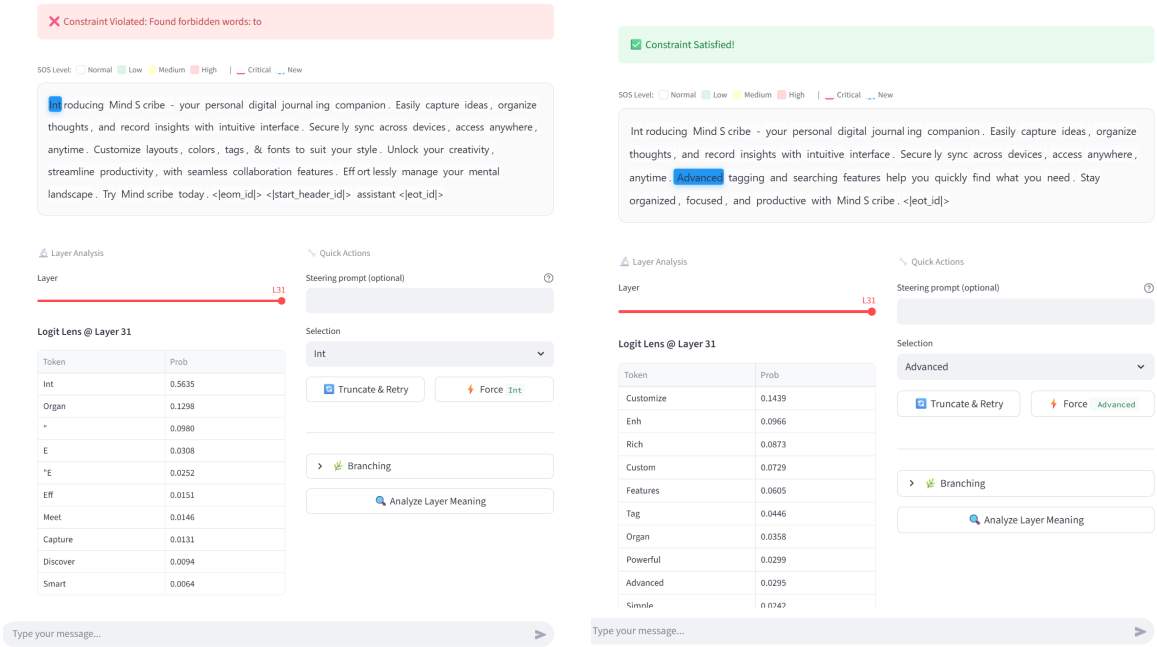

Figure 5: Alternative repair trajectory. The participant uses checker feedback and interface state to decide where to intervene before continuing generation.

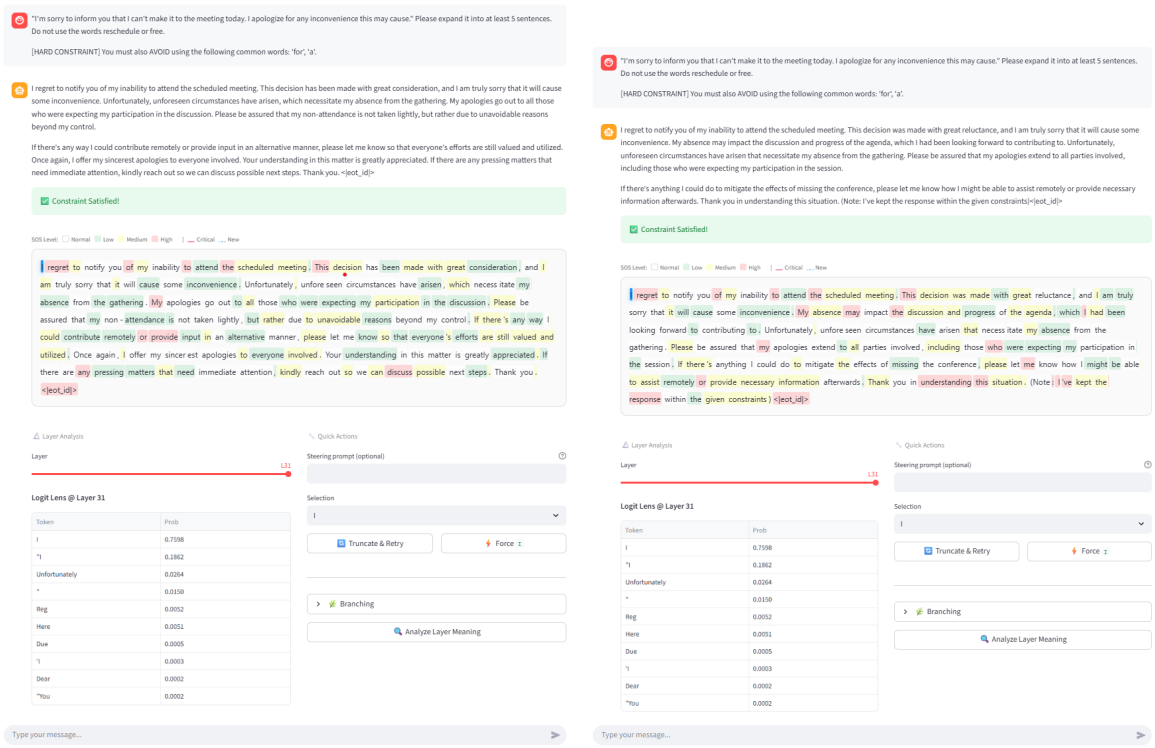

Figure 6: Examples of diagnostic signals in the full Steer interface. The Steer-lite ablation removes these cues while preserving the same local repair affordances.

Item (1–5)	**Chat Mean (SD)**	**Steer Mean (SD)**	Δ	**Exploratory** p
Q1 Control Precision	3.00 (0.65)	4.35 (0.81)	+1.35	$< .001$
Q2 Correction Efficiency	2.20 (0.62)	4.20 (0.89)	+2.00	$< .001$
Q3 Result Prediction	1.90 (0.55)	3.40 (0.50)	+1.50	$< .001$
Q4 Guidance Helpfulness	2.70 (0.92)	3.80 (0.89)	+1.10	$< .001$
Q5 Interaction Effort	3.50 (1.24)	3.65 (1.42)	+0.15	0.748
Q6 Logic Understanding	3.15 (1.09)	3.65 (0.81)	+0.50	0.163
Q7 Task Fatigue	3.80 (0.89)	3.00 (1.03)	-0.80	0.002
Q8 Emotional Experience	3.50 (0.69)	3.45 (0.76)	-0.05	0.815
Q9 Constraint Compliance	2.45 (0.69)	4.05 (0.69)	+1.60	$< .001$
Q10 Task Usefulness	2.70 (0.66)	3.60 (0.68)	+0.90	$< .001$
Q11 Ease of Use	4.55 (0.51)	3.50 (0.95)	-1.05	$< .001$
Q12 System Trust	2.90 (0.97)	4.55 (0.51)	+1.65	$< .001$
Q13 Willingness to Use	4.20 (0.89)	4.35 (0.81)	+0.15	0.330
Q14 Error Localization	2.35 (0.49)	4.15 (0.49)	+1.80	$< .001$
Q15 Repair Confidence	2.30 (0.66)	4.25 (0.64)	+1.95	$< .001$
Q16 Signal Interpretability	2.40 (0.50)	3.65 (0.49)	+1.25	$< .001$

Table 6: Item-level statistics for the 1–5 Likert survey. Δ is Steer minus Chat.

Overall Item (1–7)	**Chat Mean (SD)**	**Steer Mean (SD)**	Δ	**Exploratory** p
Task Completion Effectiveness	5.10 (0.85)	5.65 (0.49)	+0.55	0.017
Process Fluency	5.35 (0.88)	5.05 (0.76)	-0.30	0.030
Sense of Control	5.35 (0.99)	6.10 (0.55)	+0.75	0.007
Overall Satisfaction	4.50 (1.15)	5.00 (0.79)	+0.50	0.056
Future Usage Tendency	4.70 (1.03)	5.00 (0.65)	+0.30	0.209

Table 7: Overall comparison items on a 1–7 scale. Δ is Steer minus Chat.

1 User Experience Survey: Prompt-with-Steer vs. Chat

1.1 Instructions

Participants are asked to evaluate two interaction modes based on their experience:

- **Mode A: Traditional Chat** (Conversational free interaction)
- **Mode B: Prompt-with-Steer** (Controllable segmented generation)

Please rate each mode on a 5-point Likert scale *(1 = Strongly Disagree/Poor, 5 = Strongly Agree/Excellent)*. Unless otherwise specified, please answer based on your experience **in the tasks of this study** (Taboo forbidden words / JSON formatting) rather than general long-term use.

1.2 Detailed Evaluation

Table 1: User Experience Questionnaire Items. Participants rated both modes side-by-side.

ID	Assessment Item & Anchors	Chat Mode					Steer Mode				
Dimension 1: Control & Efficiency											
1	**Control Precision** I can precisely control the model to output what I want. (1: Hard to control ↔ 5: Precise control)	1	2	3	4	5	1	2	3	4	5
2	**Correction Efficiency** When I find an error, I can fix it quickly. (1: Hard to fix ↔ 5: Quick fix)	1	2	3	4	5	1	2	3	4	5
3	**Result Prediction** I can predict the model's next output. (1: Unexpected ↔ 5: As expected)	1	2	3	4	5	1	2	3	4	5
4	**Guidance Helpfulness** The system provided helpful signals to guide my actions. (1: Useless/None ↔ 5: Very helpful)	1	2	3	4	5	1	2	3	4	5
Dimension 2: Cognitive Load											
5	**Interaction Effort** Figuring out what to do next to fix an error required little trial-and-error. (1: Lots of trial-and-error ↔ 5: Very straightforward)	1	2	3	4	5	1	2	3	4	5
6	**Logic Understanding** I can understand the model's generation logic. (1: Black box ↔ 5: Clear logic)	1	2	3	4	5	1	2	3	4	5
7	**Task Fatigue** After completing the tasks in this mode, I felt mentally relaxed (not exhausted). (1: Exhausted ↔ 5: Relaxed)	1	2	3	4	5	1	2	3	4	5
8	**Emotional Experience** I felt calm and not frustrated during the task. (1: Frustrated ↔ 5: Calm/Positive)	1	2	3	4	5	1	2	3	4	5
Dimension 3: Output Quality											
9	**Perceived Constraint Compliance** I am confident the output complied with the task constraints (format / forbidden words). (1: Not confident ↔ 5: Very confident)	1	2	3	4	5	1	2	3	4	5

Figure 7: User Experience Survey (Page 1)

ID	Assessment Item & Anchors	Chat Mode					Steer Mode				
10	**Task Usefulness** The output was useful for completing the task (beyond merely passing the checker). (1: Not useful ↔ 5: Very useful)	1	2	3	4	5	1	2	3	4	5
Dimension 4: Overall Experience											
11	**Ease of Use** The system is easy to learn and use. (1: Hard to learn ↔ 5: Easy to use)	1	2	3	4	5	1	2	3	4	5
12	**System Trust** I trust the system's output to be correct. (1: Distrust ↔ 5: Fully trust)	1	2	3	4	5	1	2	3	4	5
13	**Willingness to Use** I am willing to use this interaction mode long-term. (1: Not willing ↔ 5: Very willing)	1	2	3	4	5	1	2	3	4	5
Dimension 5: Transparency & Repair											
14	**Error Localization** When something went wrong, I could quickly locate *where* the error started (which token/segment). (1: Hard to locate ↔ 5: Easy to locate)	1	2	3	4	5	1	2	3	4	5
15	**Repair Confidence** I felt confident that my intervention (editing / rollback / token choice) would improve the outcome. (1: Not confident ↔ 5: Very confident)	1	2	3	4	5	1	2	3	4	5
16	**Signal Interpretability** I could understand what the system signals meant (e.g., uncertainty/diagnostic indicators) and how to act on them. (1: Hard to understand ↔ 5: Easy to understand)	1	2	3	4	5	1	2	3	4	5

1.3 Direct Comparison

Which mode do you prefer?
(7-point scale; 1 = Strongly Prefer Chat, 4 = Neutral, 7 = Strongly Prefer Steer)

Comparison Aspect	Preference (Chat ↔ Steer)						
1. Effectiveness of task completion	1	2	3	**4**	5	6	7
2. Fluency of the process	1	2	3	**4**	5	6	7
3. Sense of control over output	1	2	3	**4**	5	6	7
4. Overall satisfaction	1	2	3	**4**	5	6	7
5. Future usage tendency	1	2	3	**4**	5	6	7

1.4 Open-ended Feedback

- **Pros of Steer Mode:** _____
- **Cons of Steer Mode:** _____
- **Suggestions:** _____

Figure 8: User Experience Survey (Page 2)