From Words to Widgets for Controllable LLM Generation

Chao Zhang*
cz468@cornell.edu
Cornell University
Ithaca, NY, USA

Yiren Liu*
yirenl2@illinois.edu
University of Illinois Urbana-Champaign
Champaign, IL, USA

Lunyiu Nie*
lynie@utexas.edu
The University of Texas at Austin
Austin, TX, USA

Jeffrey M. Rzeszotarski
jrzeszotarski@loyola.edu
Loyola University Maryland
Baltimore, MD, USA

Yun Huang
yunhuang@illinois.edu
University of Illinois Urbana-Champaign
Champaign, IL, USA

Tal August
taugust@illinois.edu
University of Illinois Urbana-Champaign
Champaign, IL, USA

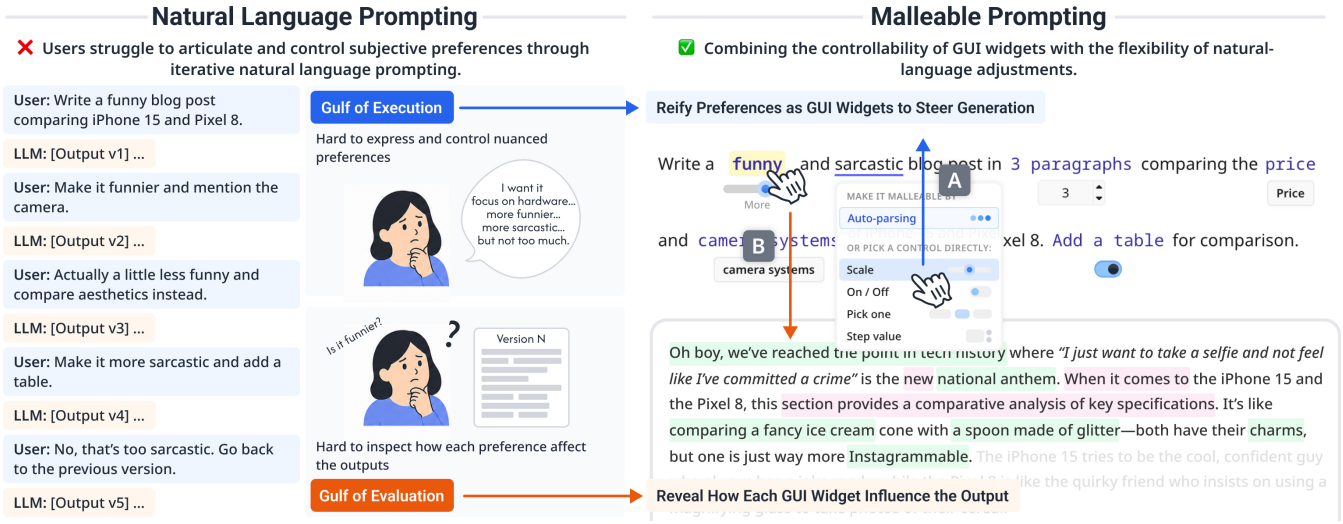

Figure 1: *Malleable Prompting.* Instead of repeatedly refining outputs through underspecified natural-language prompts (left), Malleable Prompting lets users reify prompt preferences as GUI widgets and directly manipulate them to steer generation (A). Hovering over a widget reveals the output spans influenced by that preference, making individual effects inspectable (B).

Abstract

Natural language remains the predominant way people interact with large language models (LLMs). However, users often struggle to precisely express and control subjective preferences (e.g., tone, style, and emphasis) through prompting. We propose Malleable Prompting, a new interactive prompting technique for controllable LLM generation. It reifies preference expressions in natural language prompts into GUI widgets (e.g., sliders, dropdowns, and toggles) that users can directly configure to steer generation, while visualizing each control's influence on the output to support attribution and comparison across iterations. To enable this interaction, we introduce an LLM decoding algorithm that modulates the token probability distribution during generation based on preference expressions and their widget values. Through a user study, we show that Malleable Prompting helps participants achieve target preferences more precisely and is perceived as more controllable and transparent than natural language prompting alone.

CCS Concepts

• **Human-centered computing** → **Natural language interfaces**;
• **Computing methodologies** → **Natural language processing**.

Keywords

Prompting, Large Language Models, Human-AI Interaction

ACM Reference Format:
Chao Zhang, Yiren Liu, Lunyiu Nie, Jeffrey M. Rzeszotarski, Yun Huang, and Tal August. 2026. From Words to Widgets for Controllable LLM Generation. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3830398.3830587

*These authors contributed equally to this work.

1 Introduction

When interacting with large language models (LLMs), users provide instructions via prompts, yet models rarely generate content that aligns perfectly with user intent on the first try. As a result, users often engage in an iterative process [35], refining the model's output through turn-by-turn conversations to better match their preferences. For example, a user may ask a model to draft a blog post comparing two phones, then iterate by specifying comparison aspects (e.g., price, battery, aesthetics), altering the structure (bulleted vs. narrative), and fine-tuning style (e.g., "more concise," "less formal"). While prior work has helped users revise concrete aspects of generated text, such as replacing words with synonyms, through direct manipulation [33], users still struggle to articulate and control more subjective *preferences*—such as tone, style, and emphasis—through prompting alone [26, 46]. This challenge stems from two interaction gulfs in natural language (NL) prompting:

- First, there is a *gulf of execution* [19]: users may not have clear preferences in mind at the outset, and articulating nuanced preferences in language is notoriously difficult [51]. Even when users do have a preference, it is still difficult for them to align model output with their expectations for the degree to which that preference is expressed, relying on vague modifiers ("more concise," "not that concise") to communicate subtle adjustments.
- Second, there is a *gulf of evaluation* [36]: as specified preferences accumulate across turns, the linear chat history grows long and unwieldy to navigate. Users struggle to compare iterations or attribute differences in the output to particular instructions, making it difficult to decide what to do next [14, 56].

Graphical user interfaces (GUIs) [20] have long helped reduce the interaction gulfs by turning users' goals into controllable widgets (e.g., dropdowns, toggles, and sliders). In human-LLM interaction, prior work has used widgets to make prompts more structured and reusable [2, 17, 29, 48]. However, these systems typically allow users to adjust only predefined dimensions, such as the intensity of a particular attribute using sliders [7], and provide limited support for inspecting how a given widget influences the generated output. In this paper, we introduce a more flexible and transparent interaction technique that jointly mitigates these two gulfs in preference control by combining the controllability of GUI widgets with the flexibility of natural-language adjustments (Fig. 1). Specifically, we make the following contributions:

(1) We propose Malleable Prompting (§3), a novel interaction technique that allows users to reify arbitrary preference expressions in their prompts into diverse GUI widgets (e.g., toggles, dropdowns, steppers, and sliders). With Malleable Prompting, users can directly manipulate preferences in their prompts through widgets to control generation (addressing the gulf of execution) and inspect how each preference/widget shapes the output (addressing the gulf of evaluation).
(2) We build an example system (§4) that instantiates Malleable Prompting. This system automatically detects and transforms preference expressions in a prompt into GUI widgets, enabling users to configure these widgets to steer generation, inspect their effects on the output via highlights, and track iterations along with widget configurations through a node-link diagram.
(3) We introduce a new LLM decoding algorithm that enables Malleable Prompting (§5). This technical approach modularizes prompts into controllable attributes, enabling independent control over specific preferences to steer model generation and the attribution of generated spans to their corresponding widgets.
(4) We conduct a user study (§6) comparing our method with NL prompting alone on controlled iterative content-creation tasks ($N = 12$). The results showed that our approach helped users more precisely achieve the target preferences and that they perceived it as more controllable and transparent.

In §7, we discuss how Malleable Prompting addresses the two gulfs and explore its broader applications beyond the example system presented in this paper.

2 Background and Related Work

2.1 Challenges in Iterative Prompting

Prompting is expressive: users can describe goals, constraints, and preferences in everyday language, making LLMs easy to access for open-ended tasks such as information search, summarization, and writing [4, 12, 52–55]. However, natural language is also an ambiguous interaction medium [11, 13, 28, 31]. Because many intents and preferences are under-specified, subjective, or context-dependent, users often struggle to convey what they mean in a way the model reliably interprets [51], especially in cases of nuanced preferences like tone and style. Thus users are often left repeatedly calibrating prompts to approximate an intended target through trial-and-error probing [35]. These difficulties are compounded by the turn-by-turn conversational interface common in LLM interaction. Iterative prompting scatters preferences across multiple messages, leading to challenges in attribution across drafts [14, 24, 56], even with conversation histories [14] due to lack of reusability.

In order to mitigate these challenges, prior work has designed various tools to aid users during prompt tuning. One line of work automatically suggests effective prompts or directly optimizes user prompts from feedback or examples [8, 16, 27]. Other works build visual environments for prompt development, such as prompt IDEs that support systematic experimentation (e.g., organizing variants, running batches, and comparing outputs) [3, 23, 34, 45], as well as visual programming interfaces that help users compose multi-step LLM workflows and inspect intermediate results [49, 50, 57]. Overall, these systems strengthen the *engineering* aspect of NL prompting, but they often still treat user preferences as free-form text that must be repeatedly re-specified and interpreted by the model. Given the past research, preserving the flexibility of NL prompting while giving users control over how preferences are applied motivates our work. More specifically, we focus on *controllability*: we support the exploration, adjustment, and attribution of subjective dimensions expressed in prompts, making preferences directly manipulable and their effects inspectable for users.

2.2 Blending Language and Direct Manipulation

Beyond pure NL prompting, prior work has explored hybrid interfaces that combine natural-language instructions with direct-manipulation or GUI-like controls. Many such systems use buttons, tags, or other widgets to execute predefined prompts, package

Table 1: *Comparison of Malleable Prompting with closely related systems that combine prompts with direct manipulation.* Malleable Prompting supports training-free steering of arbitrary user-defined preference spans and reveals each widget's influence through span-level attribution.

System	Domain	Steering	Attribution
Malleable Prompting	Text	Training-free; arbitrary spans	Span-level
PromptCanvas [2]	Text	✗	✗
DirectGPT [33]	Text	✗	✗
Steerable Chatbots [7]	Text	Pretrained steering vectors	✗
PromptCharm [48]	Image	Database-defined modifiers	Attention
AdaptiveSliders [21]	Image	Pretrained LoRA	✗

reusable instructions [15, 38], or populate prompt templates [29]. For example, PromptCanvas [2] enables users to compose and reuse prompt fragments (e.g., story elements) to create new prompts for revising generated content. DirectGPT compiles users' direct edits to generated text into prompt operations [33]. These systems primarily use widgets to organize, reuse, or assemble prompt text before generation. They do not generally support steering of a semantic preference during generation, nor do they reveal how a particular control influences the resulting output.

A smaller body of work uses widgets as interactive controls for steering generative models. In image generation, PromptCharm [48] supports attention-based manipulation of a database-defined set of modifiers, while AdaptiveSliders [21] uses pretrained LoRA modules to provide adjustable visual attributes. In text generation, Steerable Chatbots [7] provides a set of predefined opposing dimensions (e.g., budget/luxury and kids/adults) by pretraining steering vectors.

Together, these systems demonstrate the value of pairing language with direct manipulation. Malleable Prompting extends this direction by supporting more flexible preference control and adding an evaluative component that has received limited attention in prior text-generation systems. Specifically, rather than restricting interaction to dimensions predefined by developers, databases, or pretrained modules, Malleable Prompting allows users to turn arbitrary, context-specific preference expressions in their own prompts into a diverse set of manipulable widgets. It further provides span-level attributions that reveal how each widget influences specific parts of the generated text. This flexible and inspectable interaction is enabled by a new training-free, decoding-time steering method, rather than LoRA modules or steering vectors that require pretraining. We summarize these distinctions between Malleable Prompting and the most closely related systems in Table 1.

2.3 Steering Large Language Models

Enabling Malleable Prompting requires mechanisms that let users steer model generation and inspect outputs based on their preferences through GUI controls at interaction time. Prior steering approaches such as parameter-efficient fine-tuning (e.g., LoRA) [18] and activation steering [7] can offer strong control, but typically require additional training, curated data, or offline preparation. This limits their applicability in controlling arbitrary preference expressions. In HCI systems, the most common interaction-time strategy is instead to encode values or intensities using meta-instructions within a prompt (e.g., "on a scale from 0 to 10" in Textoshop [32]). We instead introduce a new LLM decoding algorithm that isolates and controls each attribute by directly modulating the model's next-token probability distribution during inference. This method supports (1) real-time adjustment of multiple independent attributes via GUI controls and (2) attribution by linking changes in token probabilities to the corresponding controls. We detail our technical method and compare it with prompt-based scaling in §5, showing that our method provides stronger and better-calibrated control.

3 What is Malleable Prompting?

Here, we formalize the concept of Malleable Prompting, which serves to guide the interaction design described in §4 and the technical method introduced in §5.

3.1 Preferences and Attributes

A prompt can be composed of two distinct parts: a *task specification* (the core objective) and *preference expressions* (constraints on execution) [59]. For example, in the prompt "*Write a friendly Slack announcement to team members about a new company-wide social event in a brief paragraph*," the task is the explanation itself, while "*team members*" (audience), "*brief*" (length), "*friendly*" (tone), and "*Slack announcement*" (format) act as constraints that shape *how* that task is executed. Here, we define an *attribute* as a specific preference constraint embedded in the prompt that can be parameterized. A *control* is the GUI widget that exposes this attribute for manipulation. The core problem Malleable Prompting addresses is transforming a static prompt into a set of independently manipulable attributes, exposed as controls that users can adjust interactively.

To systematically model how users constrain LLM outputs via preference expressions, we conceptualize a *preference space* based on common dimensions identified in literature [22, 25, 30, 58, 59]. We categorize these into five primary dimensions (Table 2): Format & Structure (structural organization), Tone & Style (stylistic flavor), Audience & Persona (target reader or assumed persona), Length & Conciseness (conciseness constraints), and Content Constraints (specific inclusions/exclusions). We use this taxonomy to inform the design and guide the implementation of the system (see §4). While not exhaustive, this taxonomy captures the most common preference dimensions that users actively control. Importantly, our system remains flexible enough to accommodate new, emergent preference dimensions beyond this foundational set.

To incorporate these high-level preference dimensions in an interactive technique, we must map them to specific, manipulable data structures. Drawing from established taxonomies in measurement theory [44], we classify attributes into four types based on their value properties: categorical, numeric, binary, and continuous.

- **Categorical:** A choice among mutually exclusive nominal options (e.g., output format: "HTML," "Markdown," "Plain Text").
- **Numeric:** A countable, discrete quantity (e.g., "3 paragraphs").
- **Binary:** A presence/absence constraint (e.g., "include a title" vs. none; "use bullet points" vs. none).
- **Continuous:** An ordinal intensity along a stylistic spectrum (e.g., "more formal," "less concise").

Table 2 illustrates the typical attributes through which each preference dimension manifests.

Table 2: *Taxonomy of the preference space.* The taxonomy organizes the preference space based on common dimensions identified in prior literature [22, 25, 30, 58, 59], along with the typical attribute types through which each dimension manifests.

Dimension	Description	Examples	Typical Attribute(s)	Sources
Format	How the output is visually, logically, or syntactically organized.	"as a poem," "in JSON," "use bullet points," "two columns"	Categorical, Binary	[22, 25, 30, 58, 59]
Tone	The emotional resonance, voice, or stylistic flavor of the writing.	"funny," "professional," "academic," "enthusiastic"	Categorical, Continuous	[22, 25, 30]
Audience	Who the text is tailored for, or the role the LLM should assume.	"explain to a 5-year-old," "for domain experts," "act as a lawyer"	Categorical, Binary	[25, 30, 58, 59]
Length	Constraints on the physical size or detail level of the output.	"short," "under 200 words," "3 paragraphs," "concise"	Numeric, Continuous	[25, 30, 58]
Content	Specific rules about what information to include or exclude.	"no jargon," "include references," "focus on the impact"	Categorical, Binary	[30, 58]

3.2 Problem Formulation

Formally, let P_0 represent the initial task specification, $\mathcal{A}$ represent the set of attributes for preference expressions, partitioned into four subsets: $\mathcal{A}^{\text{cat}}$, $\mathcal{A}^{\text{num}}$, $\mathcal{A}^{\text{bin}}$, and $\mathcal{A}^{\text{cont}}$. We model the full prompt P as the composition of P_0 with these attribute sets:

$$P_{\mathcal{A}} = P_0 \oplus (\mathcal{A}^{\text{cat}} \cup \mathcal{A}^{\text{num}} \cup \mathcal{A}^{\text{bin}} \cup \mathcal{A}^{\text{cont}}) \quad (1)$$

Here, $\oplus$ denotes a composition operator that integrates attributes into a parameterized prompt. Each attribute corresponds to a phrase span in the prompt. Discrete attributes (i.e., categorical, numeric, binary) take on distinct values, while continuous attributes represent intensity along a semantic scale (e.g., "concise"). Given a parameterized prompt $P_{\mathcal{A}}$, generation produces an output $Y \sim p(\cdot \mid P_{\mathcal{A}})$. Malleable Prompting aims to support interactive exploration over $\mathcal{A}$ by enabling users to vary attributes and observe the corresponding changes in Y.

3.3 Design Goals

To bridge the gulfs of execution and evaluation identified in §1, and to operationalize the formal definition of Malleable Prompting into an interactive system, we derive two core design goals (DG):

DG1 (Execution): Reify Preferences as GUI Widgets to Steer Generation: To realize the model of $P_{\mathcal{A}}$, the system should be able to help users identify controllable preferences embedded in a user's prompt (P) and reify them into appropriate GUI widgets (elements of $\mathcal{A}$). Moreover, because users often do not know their full set of preferences a priori and preferences emerge in response to seeing outputs [51], the system should allow users to add new constraints that expand $\mathcal{A}$ over time and can suggest relevant unstated preferences.

DG2 (Evaluation): Reveal Widget Effects on Output to Support Attribution: In standard chat interfaces, the link between a prompt tweak and the resulting output change is often opaque. To support the evaluation of outputs (Y) so users can make informed iterations, the system must explicitly map the relationship between control adjustments and text variations. It should help users track how specific attribute values influence specific text spans and organize drafts around configurations (states of $\mathcal{A}$) rather than linear conversation history, enabling users to easily compare, revert, and branch iterations.

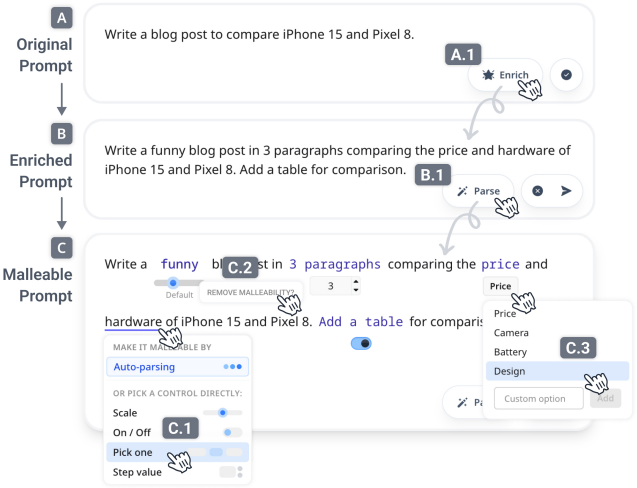

Figure 2: *Making prompts malleable.* Users can optionally enrich an initial prompt with suggested preferences (A.1), automatically parse controllable attributes and convert them into widgets or manually bind arbitrary selected text to a widget (C.1). Users can remove widgets via "Remove Malleability" (C.2). For categorical attributes, the system also suggests alternative values for later iterations (C.3).

4 System Design

Malleable Prompting is an interactive prompting technique that allows users to reify preference expressions in their prompts into embedded GUI controls. With Malleable Prompting, users can directly manipulate these preferences through widgets to steer generation (**DG1**) and inspect how each preference shapes the output (**DG2**). To illustrate this, we instantiated Malleable Prompting within a research prototype guided by the two design goals and established principles of direct manipulation [19, 33, 39]. We discuss broader applications of Malleable Prompting across several other interactive systems in §7.3. In this section, we detail the prototype's core functionalities through a user scenario.

4.1 User Scenario

Assume a user, Alice, wants to draft and iteratively refine a blog post using an LLM. To begin, she enters the prompt: "*Write a blog post in 3 paragraphs comparing iPhone 15 and Pixel 8.*" into the system.

Enriching Prompts with Suggested Preferences: Initially, Alice is unsure of the specific style and constraints for her post. She clicks the star ★ icon (Fig. 2A.1), and the system automatically enriches her initial prompt with several preference expressions to help her get started (**DG1**). The prompt becomes: "*Write a funny blog post in 3 paragraphs comparing the price and hardware of iPhone 15 and Pixel 8. Add a table for comparison.*" Alice decides to proceed with this enriched version.

Parsing Prompts into Controllable Widgets: Next, Alice clicks the auto-parsing icon (Fig. 2B.1). The system automatically extracts controllable attributes from the text and reifies them into GUI controls (**DG1**). It converts "*funny*" into a continuous tone slider, "*3 paragraphs*" into a numeric length stepper, "*price and hardware*" into a categorical dropdown, and "*Add a table for comparison*" into a binary formatting toggle. Alice also notices that she can manually highlight any text span to bind it to a specific widget type: slider, toggle, dropdown, or stepper (**DG1**). She replaces the auto-generated dropdown for "*price and hardware*" with two distinct categorical dropdowns: one for "*price*" and another for "*hardware*" (Fig. 2C.1). For these controls, the system suggests alternative values (e.g., *aesthetics, camera, battery*) to populate the dropdown menus, which Alice can easily include or discard.

Steering Generation via Widget Manipulation: With the controls configured, Alice explores different preference states (**DG1**). She shifts the tone slider toward a higher intensity of "*funny*," leaves the table toggle enabled, changes "*hardware*" to the more specific "*camera*" option, and changes the comparison dimension from "*price*" to "*design*" via the dropdowns (Fig. 2C.2). Upon clicking the generate icon, the system leverages this specific attribute configuration to steer the LLM, producing a draft that adopts a humorous tone and formats the design and hardware comparison as a table.

Inspecting Widget Influence on the Output: To understand exactly how her configuration shaped the draft, Alice clicks "*funny*" to inspect its specific effects (**DG2**). The system responds by highlighting the impacted sentences in the output. Within those sentences, specific words and phrases encouraged by "*funny*" (e.g., "*Oh boy,*" "*national anthem,*" "*a spoon made of glitter*") are highlighted with a green background, while discouraged formal or dry terms are marked with a red background. She similarly inspects the "*camera*" dropdown and the formatting toggle, quickly verifying how those specific constraints localized their impact within the text. By exploring these granular highlights across different widgets, Alice can make informed decisions about whether her current configuration aligns with her intent for the next iteration.

Iterating Content via Widget Configuration: Based on these highlights, Alice refines her content by configuring her malleable prompts. She types a new preference expression "*sarcastic*" into the text and converts it into a new continuous slider control (Fig. 1) to pivot the blog post's overall flavor (**DG1**). Accordingly, she decreases the intensity of the "*funny*" tone slider to better balance the two comedic styles. Over the next few minutes, Alice configures her malleable prompt multiple times, tweaking slider intensities, adjusting the number of paragraphs, toggling the comparison table, and swapping dropdown values to generate and evaluate several alternative versions of the blog post.

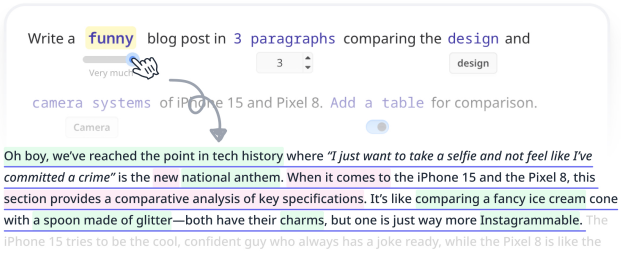

Figure 3: *Inspecting widget effects on generated text.* Clicking a widget reveals its localized influence on the output: affected sentences are underlined, positively induced phrases are highlighted in green, and discouraged wording is highlighted in red.

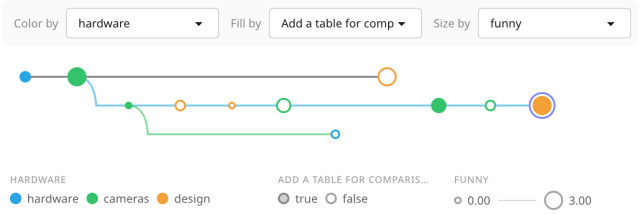

Figure 4: *Tracking prompt iterations with a node-link graph.* Versions generated with the same prompts and widgets appear along the same horizontal branch. Adding new preferences or widgets creates a new vertical branch. Visual encodings map widget values to node color for categorical attributes, node fill for binary attributes, and node size for continuous or numerical attributes.

Managing Iterations via the Version Graph: After several rounds of iterations, Alice uses the version graph visualization to make sense of her iteration history (**DG2**). The graph displays her generated alternatives as a node-link diagram, arranged chronologically from left to right (Fig. 4). Nodes representing versions generated with the same prompts and widgets, though possibly with different values, appear on the same horizontal branch. Conversely, adding new preferences and widgets (like the "*sarcastic*" slider) creates a new vertical branch. Different attributes can be mapped to different visual encoding channels. For example, Alice maps the categorical "*hardware*" dimension to node color, the binary "*table*" constraint to node fill (solid vs. hollow), and the continuous "*funny*" tone to node size. She can click any node to revert the prompt to that exact state. By comparing prompt configurations and their corresponding outputs, Alice identifies her preferred version and exports it.

4.2 Implementation Notes

The frontend of our system is built using Next.js. The core technical method (described in §5), responsible for modularizing prompts into controllable attributes for manipulation and attribution, and steering a locally deployed language model's generation during decoding, is implemented in Python. Other text processing components, including enriching prompts, parsing attributes, and generating options, are powered by prompting the `GPT-5.3` model via the OpenAI API. Prompts used are provided in Appendix B.

5 Technical Methods

In this section, we present the technical method underlying Malleable Prompting for addressing the problem formulated in §3.2. The interaction in Malleable Prompting requires the ability to steer model generation according to customized widgets and their values, for arbitrary preference expressions selected from the prompt. This requirement makes many steering approaches used in prior HCI systems infeasible, such as parameter-efficient fine-tuning methods [18, 21] and activation steering [7], because they typically rely on additional training, curated data, or offline preparation. Instead, our method builds on the observation that adding or removing a phrase changes token probability distribution during inference [17], and that these changes can be leveraged to steer generation at decoding time.

Based on this insight, we introduce a training-free method with two key components: (1) steering generation based on user-adjusted configurations for each attribute, and (2) attributing the effect of each attribute to corresponding textual spans in the output.

5.1 Prompt Modularization

In this component, we first instruct `GPT-5.3` to extract controllable attributes $\mathcal{A}$ in a given prompt P and the corresponding phrase span (preference expressions) associated with each attribute. We then split P into an initial task specification P_0 (e.g., "*write an email ...*") and a set of attributes $\mathcal{A}$ (e.g., "*in a formal tone*" and "*using bullet points*") for preference expressions.

For each attribute $a \in \mathcal{A}$, we isolate its specific contribution to the generation. At each decoding step i, we estimate the token-level influence of a as the difference in log-probabilities between prompting with and without the attribute:

$$F_{a,P_0}(x_i) = \log p(x_i \mid x_{<i}, a, P_0) - \log p(x_i \mid x_{<i}, \varnothing, P_0). \quad (2)$$

The value $F_{a,P_0}(x_i)$ quantifies the degree to which including a shifts the likelihood of token x_i in the current context.

For binary and continuous attributes, we expose a scalar control λ_a that scales the attribute's intensity during decoding:

$$\log p_{\lambda_a}(x_i \mid x_{<i}, a, P_0) = \log p(x_i \mid x_{<i}, \varnothing, P_0) + \lambda_a F_{a,P_0}(x_i). \quad (3)$$

By default, $\lambda_a = 1$. Setting $\lambda_a = 0$ recovers the original distribution without the attribute, while varying λ_a amplifies or attenuates its effect. We restrict $\lambda_a \in \{1, 0\}$ for binary attributes to emulate an on/off toggle. For continuous attributes, we map $\lambda_a \in [0, 3]$ to a 7-point slider with increments of 0.5. We selected this range based on the technical evaluation described below. Appendix C presents example outputs generated with different values of λ for continuous attributes. Conversely, categorical and numeric attributes represent discrete constraints that require strict adherence, such as a specific language or word count. We handle these by enforcing the user's choice through direct string substitution of a in P.

To account for the combined effect of all attributes, we define the final modulated distribution. Let $\mathcal{A}_{\text{sub}} = \mathcal{A}^{\text{cat}} \cup \mathcal{A}^{\text{num}}$ be the set of attributes handled via substitution, and $\mathcal{A}_{\text{mod}} = \mathcal{A}^{\text{bin}} \cup \mathcal{A}^{\text{cont}}$ be those handled via modulation. We combine their effects as follows:

$$\log p_{\boldsymbol{\lambda}}(x_i \mid x_{<i}, \mathcal{A}) = \log p(x_i \mid x_{<i}, P_{\text{sub}}) + \sum_{a \in \mathcal{A}_{\text{mod}}} \lambda_a F_{a,P_0}(x_i), \quad (4)$$

where P_{sub} represents the prompt formed by substituting all attributes of $\mathcal{A}_{\text{sub}}$ in P_0 with their user-selected values.

This method allows users to steer model generation with their chosen preferences at decoding time without offline preparation or additional training. We implement it using an open-source model (`Qwen3-8B`) to enable access to logits and custom decoding.

Technical Evaluation: We evaluated whether our method provides (1) stronger and (2) better-calibrated control than prompt-based scaling using an LLM-as-a-judge protocol with human validation. For the baseline, we prepended explicit numeric instructions for each active attribute to control intensity (e.g., "*Apply the following writing style settings: formality: 2/7*," following Masson et al. [32]) and generated outputs using standard decoding. For our method, we explored $\lambda \in [-10, 10]$. Using 100 prompts sampled from the $\texttt{WildChat-WC}_{\texttt{Wr}}$ dataset [35] and 30 preference combinations created based on the corpus of Lee et al. [25], we varied the baseline's numeric intensity settings from 1 to 7 and mapped λ to the same 1–7 scale to generate outputs. We then used an LLM judge (`GPT-5-mini`) to rate the perceived intensity of each attribute in each output on a 1–7 scale. We validated the judge on 126 anonymized outputs, each of which was rated by three authors, and found moderate agreement with the human annotations ($\kappa = 0.410$, $p < .001$).

For each method and dimension, we fit a least-squares regression of the judge score on the intensity level, where the regression coefficient β estimates the expected change in perceived attribute intensity for each one-unit increase. Overall, our steering method achieved greater steering magnitude than the prompt-only baseline across the tested attributes (mean $\beta = 0.61$ vs. 0.32, $p < .001^{***}$). Moreover, our method produced slopes significantly closer to 1 than the prompt-only baseline (mean $|\beta - 1| = 0.40$ vs. 0.69; paired $t = -5.59$, $p < .001^{***}$), indicating that it more closely approximated the intended intensity scale. Examining the full λ sweep further revealed that steering strength increased substantially until approximately $\lambda = 3.33$, after which performance saturated. Also considering negative λ values were difficult to interpret as user-facing control settings, we therefore map the slider in the final deployed system to a λ range of $[0, 3]$ by rounding the empirically identified saturation point of 3.33 down to 3. We provide additional details of the protocol and results in Appendix D.

5.2 Prompt Attribution

The second component traces which portions of the generated output are influenced by each controllable attribute. We adopt a post-hoc attribution method that provides users with transparency, allowing them to understand the specific impact of their adjustments.

For discrete attributes handled via substitution in $\mathcal{A}_{\text{sub}}$, we utilize `GPT-5.3` to identify exact, verbatim substrings within the generated output that correspond to the substituted values in P_{sub}. This lexical matching provides a direct link between the user's choice and the resulting text. In contrast, attributing influence to continuous attributes in $\mathcal{A}_{\text{mod}}$ is more complex because their effects often interact during decoding; consequently, isolating token-level influence to a single attribute can be complex. Following prior work on model explanation [10, 47], we thus compute the Shapley attribution that measures the *marginal* effect of an attribute a while accounting for the presence of others.

For each generated token x_i, we define the attribution $\phi_a(x_i)$ of attribute $a \in \mathcal{A}_{\text{mod}}$ as the ratio of its probability under the full control vector $\boldsymbol{\lambda}$ versus a counterfactual that removes only that specific attribute:

$$\phi_a(x_i) := \frac{p_{\boldsymbol{\lambda}}(x_i \mid x_{<i}, \mathcal{A})}{p_{\boldsymbol{\lambda}_{\setminus a}}(x_i \mid x_{<i}, \mathcal{A})}. \tag{5}$$

In this context, $\boldsymbol{\lambda}_{\setminus a}$ denotes the control vector where the weight for attribute a is set to zero ($\lambda_a = 0$), while all other weights remain at their user-specified levels.

A value of $\phi_a(x_i) > 1$ indicates that attribute a increases the likelihood of token x_i given the context; conversely, $\phi_a(x_i) < 1$ indicates suppression. We visualize contiguous spans of tokens with consistent positive or negative values to help users anticipate how further adjustments to λ_a will reshape the output. This approach implements the core Shapley principle of marginal contribution while avoiding exponential subset enumeration, which ensures the computation remains feasible during real-time decoding (Appendix A).

Technical Evaluation: We evaluated whether attribution highlights faithfully reflect a slider's effect by testing whether a token's displayed label predicts the direction of its probability change when the corresponding slider is adjusted. Using the same model and dataset as in §5.1, we treated intensity level 5 as the reference condition and measured token-level probability changes when varying the λ value of an individual attribute to produce higher intensity (7) or lower intensity (3 and 1).

The results showed that the tokens highlighted by our method matched the observed direction of change 81.4% of the time, compared with 67.2% for neutral tokens, indicating that the highlights carry meaningful predictive signal. This effect was also attribute-specific: agreement was 81.0% when adjusting the intensity of a token's corresponding attribute, but only 60.8% when adjusting a different attribute, near the neutral baseline. Together, these results suggest that, although our highlights are post-hoc indicators rather than causal explanations, they provide a useful widget-specific signal of how manipulating a widget can change the output. We report the full protocol and results in Appendix E and Table 7.

6 User Study

We conducted a within-subjects study with 12 participants, comparing Malleable Prompting with NL prompting. The study aimed to address the following research questions:

- **RQ1** Does Malleable Prompting help users better control LLM generation according to their preferences?
- **RQ2** Does Malleable Prompting help users better inspect the effects of preference changes on the output?
- **RQ3** How do users appropriate Malleable Prompting, and what strategies and challenges emerge?

6.1 Participants

We recruited 12 participants (7 female, 5 male) aged 24–37 (M = 26.67, SD = 3.68) via social networks and word of mouth. All participants reported using LLM tools (e.g., ChatGPT, Claude, or Gemini) for writing and editing tasks on a daily basis. Participants were generally experienced with prompting, with a mean self-rated expertise of 5.17 (SD = 0.58) on a 7-point Likert scale (1 = Not at all, 7 = Very experienced). Each participant was compensated with a $20 gift card per hour. Appendix F.1 provides detailed participant information.

6.2 Study Design

In this section, we describe the baseline and tasks.

Baseline: The baseline resembles common conversational LLM interfaces such as ChatGPT. In the baseline, users relied on NL prompting alone to generate outputs through a turn-by-turn conversation. We used the same underlying model (`Qwen3-8B`) for output generation as in our system to ensure a fair comparison. Example screenshots of the baseline can be found in Appendix F.2.

Tasks: The study consisted of two parts: a Jeopardy evaluation and a free-form exploration. The *Jeopardy evaluation*, proposed by Gao et al. [13] and adopted by subsequent work [11, 28, 31, 41–43], is a well-established methodology for evaluating natural language interfaces in settings where users must explore and refine preferences. Its core idea is to provide participants with a task and a concrete target output (a goal state), then ask them to use the system to reproduce the target as closely as possible. This design avoids a common challenge in open-ended writing studies: if participants are given only a vague instruction (e.g., "write an email requesting a meeting") without any specific goals, they may stop after a basic first draft, making it difficult to compare interfaces on iterative behavior.

For Jeopardy evaluation, we designed two writing tasks: an email task and an explanation task. These tasks reflect a common writing scenario in which we must tailor content to different recipients or audiences [5, 6]. For each task, we prepared three target versions that differed along multiple preference dimensions based on our taxonomy of the preference space (Table 2), while preserving the same underlying intent and factual content. To ensure fairness between conditions, we defined each target version using an explicit rubric of preference settings (e.g., audience = manager, tone = formal, format = bullet-point plan) and then authored the final target texts to satisfy the rubric with the help of the baseline. We iteratively refined targets to ensure that (1) successive versions were clearly distinguishable, (2) the intended preference differences were salient, and (3) each target was feasible to reproduce using either interface. All targets were fixed prior to the study and held constant across conditions. Appendix F.3 shows the tasks and target versions.

In addition to goal-directed Jeopardy tasks, we included a free-form exploration phase to capture more naturalistic behaviors that may not arise under explicit target matching. This phase was designed to surface how participants appropriate the interfaces, what strategies they adopt for iterative refinement, and what breakdowns occur in self-directed writing workflows. We employed a think-aloud protocol during this phase, followed by a semi-structured interview. Participants used the system for a diverse range of genres, including academic introductions, homework essays, slide scripts, argumentative essays, social media posts, blog posts, and creative writing such as poetry as shown in Appendix F.1.

6.3 Study Procedure

The study began with informed consent and a demographics questionnaire. Participants then completed the two parts of the study. Each session lasted approximately 90 minutes.

Jeopardy Evaluation: In each condition, participants completed one Jeopardy task (either the email task or the explanation task). At the start of each condition, participants received a brief tutorial and completed a short warm-up task to familiarize themselves with the interface (5 minutes). For the assigned task, participants were asked to produce an output matching Version 1, then revise it to match Version 2, and finally revise it again to match Version 3. To enable a controlled evaluation of the prompting methods, the Jeopardy task asks users to complete tasks using only prompting, without directly editing the output. We allotted 15 minutes for the Jeopardy task in each condition and allowed participants to generate as many drafts as needed within the time limit using NL prompting in the baseline or using Malleable Prompting. The order of systems and tasks was counterbalanced across participants. After each condition, participants completed a short survey (2–3 minutes) including 11 items (6 derived from design goals in §3.3 and 5 from Wu et al. [50]) assessing user experience.

Free-form Exploration: After completing the two Jeopardy tasks, participants completed a time-boxed free-form writing session (15 minutes) with our prototype. Participants worked on a self-chosen writing goal based on their recent real-life writing needs. In this task, participants could use both prompting and direct output editing, allowing us to observe more realistically how Malleable Prompting fit into their writing practices. We encouraged participants to think aloud throughout. After the free-form exploration, we conducted a short semi-structured interview (15 minutes) for their subjective feedback and interface-specific strategies and breakdowns (questions listed in Appendix §F.4).

6.4 Quantitative Findings

In this section, we report quantitative results from our analysis of participant event logs, task performance, and subjective ratings. For statistical analysis, we used the Wilcoxon signed-rank test with Bonferroni correction due to the small sample size and the non-normal distribution of the data.

Widget Creation: We first analyzed how participants created widgets across both tasks (Table 3). In the Jeopardy task, participants created 346 widgets, primarily sliders ($N = 181$, 52.31%), followed by toggles ($N = 86$, 24.86%), dropdowns ($N = 54$, 15.61%), and steppers ($N = 25$, 7.23%). This distribution remained consistent in the Exploration task ($N = 184$, where sliders again led at 52.17% ($N = 96$), followed by toggles ($N = 44$, 23.91%), dropdowns ($N = 38$, 20.65%), and steppers ($N = 6$, 3.26%). Overall, across both tasks, participants created a total of 530 widgets, with sliders remaining the dominant choice ($N = 277$, 52.26%).

We then categorized the selected preference expressions for each created widget based on the taxonomy of the preference space in Table 2. As shown in Table 4, the most prominent trend is the strong association between sliders and Tone. In both the Jeopardy (49.2%) and Exploration (56.2%) tasks, participants predominantly chose sliders to adjust stylistic elements, likely favoring the granular, continuous control they provide for subjective qualities. In contrast, toggles were the primary choice for Content, accounting for over 60% of toggle usage in both tasks. This suggests that participants viewed Content as binary decisions (either including or excluding specific information) which aligns naturally with the on/off functionality of toggles. Dropdowns served as the primary instrument for defining Audience, representing nearly half of all dropdown interactions (48.1% in Jeopardy and 44.7% in Exploration). This indicates that users prefer selecting from a predefined list of distinct categories when characterizing an intended audience. Steppers, while used less frequently overall, were heavily utilized for Length (e.g., 60% in the Jeopardy task) and Format, where incremental adjustments to output size or structural complexity were required. Overall, Tone (30.2%) and Content (28.1%) were the most frequently targeted categories for widget creation.

Widget Adjustment: We then examined how frequently participants adjusted the values of these widgets after creation (Table 3). In the Jeopardy task, participants performed 460 configurations, with sliders accounting for the vast majority of interactions ($N = 385$, 83.70%), followed by toggles ($N = 46$, 10.00%). This trend was even more pronounced in the Exploration task, where sliders represented 88.67% of all interactions ($N = 274$). Across both tasks, steppers and dropdowns were configured the least frequently, totaling only 2.08% and 4.03%. This indicates that while categorical widgets are typically used to establish stable constraints, sliders serve as the primary tool for iterative, granular refinement of the model's output.

Jeopardy Task Performance: In the Jeopardy evaluation, we assessed quality by evaluating whether each of the four target preferences for each version (Appendix F.3) was met. We used a two-rater blinded validation process; one rater scored all outputs against predefined binary rubrics, while a second reviewed scores and flagged ambiguous cases for discussion. Each preference dimension was evaluated independently using a binary scoring system: 1 point if the criterion was clearly satisfied based on the overall impression and dominant characteristics of the text, 0 points otherwise. This yielded a maximum score of 4 points per version. Quality scores were significantly higher in the Malleable Prompting condition ($M = 3.61, SD = 0.55$) than in the baseline ($M = 3.31, SD = 0.67$; $W = 60.50, p = .036^{*}$). This suggests that Malleable Prompting helped users achieve target preferences more precisely than NL prompting alone.

Subjective Ratings: As shown in Fig. 5, participants rated malleable prompts significantly more favorably than the baseline across both execution- and evaluation-related measures. For executing preference changes, participants rated Malleable Prompting and the baseline similarly in terms of conveying desired output preferences ($M = 6.00, SD = 0.74$ vs. $M = 4.75, SD = 1.54$; $W = 42.00, p = .141$). However, they rated Malleable Prompting higher for adjusting and steering outputs ($M = 5.92, SD = 1.08$ vs. $M = 4.00, SD = 1.60$; $W = 55.00, p = .012^{*}$), discovering new preferences ($M = 6.00, SD = 1.28$ vs. $M = 3.08, SD = 1.62$; $W = 55.00, p = .012^{*}$), and iterating closer to desired output ($M = 6.42, SD = 0.51$ vs. $M = 3.58, SD = 1.31$; $W = 66.00, p = .006^{**}$). Participants also rated Malleable Prompting significantly higher on evaluation-related abilities, including understanding cause–effect relationships between prompt changes and outputs ($M = 6.25, SD = 0.97$ vs. $M = 3.25, SD = 2.22$; $W = 55.00, p = .012^{*}$) and comparing iterations ($M = 6.58, SD = 0.51$ vs. $M = 3.08, SD = 1.83$; $W = 78.00, p = .003^{**}$). This pattern was echoed in ratings of the

Table 3: *Distribution of widget types across creation and adjustment.* This summary includes the raw counts and percentages of widget types that participants created and adjusted across both the Jeopardy evaluation and free-form exploration tasks.

Measure	Task	Slider	Toggle	Dropdown	Stepper	Total
Created	Jeopardy	181 (52.31%)	86 (24.86%)	54 (15.61%)	25 (7.23%)	346
	Exploration	96 (52.17%)	44 (23.91%)	38 (20.65%)	6 (3.26%)	184
	Total	**277 (52.26%)**	**130 (24.53%)**	**92 (17.36%)**	**31 (5.85%)**	**530**
Adjusted	Jeopardy	385 (83.70%)	46 (10.00%)	21 (4.57%)	8 (1.74%)	460
	Exploration	274 (88.67%)	17 (5.50%)	10 (3.24%)	8 (2.59%)	309
	Total	**659 (85.70%)**	**63 (8.19%)**	**31 (4.03%)**	**16 (2.08%)**	**769**

Table 4: *Distribution of widget types across preference categories.* This summary details the frequency and categorization of widget types across preference categories (Table 2) created by participants during both the Jeopardy evaluation and free-form exploration tasks.

Task	Type	Format	Tone	Audience	Length	Content	Total
Jeopardy	Slider	30 (16.6%)	89 (49.2%)	2 (1.1%)	30 (16.6%)	30 (16.6%)	181
	Toggle	28 (32.6%)	2 (2.3%)	0 (0.0%)	1 (1.2%)	55 (64.0%)	86
	Dropdown	0 (0.0%)	4 (7.4%)	26 (48.1%)	1 (1.9%)	23 (42.6%)	54
	Stepper	9 (36.0%)	0 (0.0%)	0 (0.0%)	15 (60.0%)	1 (4.0%)	25
	Subtotal	**67 (19.4%)**	**95 (27.5%)**	**28 (8.1%)**	**47 (13.6%)**	**109 (31.5%)**	**346**
Exploration	Slider	16 (16.7%)	54 (56.2%)	0 (0.0%)	23 (24.0%)	3 (3.1%)	96
	Toggle	6 (13.6%)	8 (18.2%)	2 (4.5%)	0 (0.0%)	28 (63.6%)	44
	Dropdown	11 (28.9%)	3 (7.9%)	17 (44.7%)	0 (0.0%)	7 (18.4%)	38
	Stepper	2 (33.3%)	0 (0.0%)	0 (0.0%)	2 (33.3%)	2 (33.3%)	6
	Subtotal	**35 (19.0%)**	**65 (35.3%)**	**19 (10.3%)**	**25 (13.6%)**	**40 (21.7%)**	**184**
Grand Total		**102 (19.2%)**	**160 (30.2%)**	**47 (8.9%)**	**72 (13.6%)**	**149 (28.1%)**	**530**

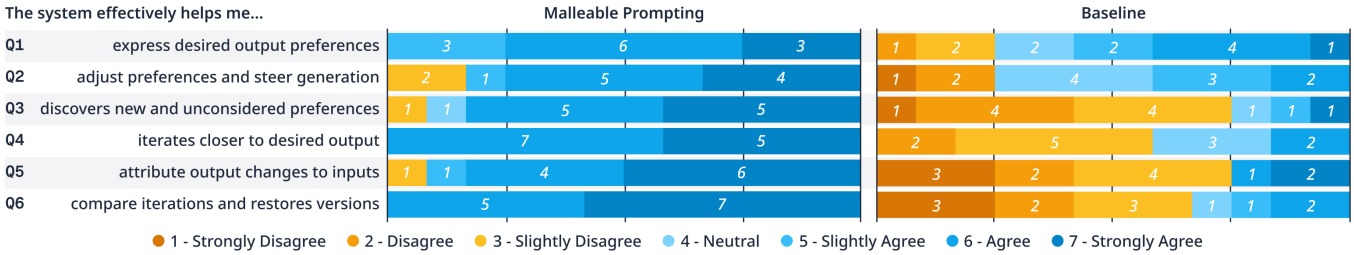

Figure 5: *Perceived support for design goals.* Participants' responses to a 7-point Likert-scale questionnaire, comparing our system against the baseline condition across both execution-related (Q1–Q4) and evaluation-related (Q5–Q6) measures.

overall experience: participants reported that Malleable Prompting better helped them think through outputs ($M = 6.42, SD = 1.00$ vs. $M = 3.67, SD = 1.67$; $W = 55.00, p = .010^{**}$), provided greater transparency ($M = 6.17, SD = 0.83$ vs. $M = 3.08, SD = 1.73$; $W = 66.00, p = .005^{**}$), and gave them a stronger sense of control ($M = 6.25, SD = 0.45$ vs. $M = 4.08, SD = 1.62$; $W = 78.00, p = .002^{**}$).

6.5 Qualitative Findings

We conducted qualitative analysis across all 12 participants' transcripts, following established thematic analysis protocols [9].

Fine-Grained Preference Control with Widgets: We found that the widgets come into play when natural language lacks the expressive resolution to communicate fine-grained stylistic preferences. Participants (11/12) described moments where NL prompting alone was difficult to convey the degree of a desired attribute. As P12 mentioned in the baseline condition, *"I want something shorter, but it gets way too short. I wanted 50%, but it compressed to 20%. I can only add more degree adverbs and try to move it back."* In contrast, P06 commented that widgets *"can achieve more possibilities than degree adverbs,"* which was echoed by five other participants. For example, P02 noted that fine-grained degree control *"is something I can't describe well with language."* P02 also pointed out that Malleable Prompting allowed them to perform more fine-grained edits over prompts: *"Each prompt becomes very modular ... widgets let me adjust many small parts of the sentence. It makes my control over this prompt very granular."* Our system's enrich feature also addressed a complementary problem: users often do not know which dimensions to control. All 12 participants valued this feature for augmenting a minimal prompt with rich specifications. P04 described this as *"... it is always more difficult to create than to revise ... Starting with something to revise is much simpler."* Users' varied language background also plays a role in this, as P06 described: *"As a non-native English speaker ... I might not even know what kind of*

result I want myself, " which is echoed by three other participants. On the other hand, we also observe some participants (5/12) specifying attributes in natural language by directly editing the input prompt when desired attributes are not present in widgets.

Evaluating Outputs with Versioning and Attribution: Participants reported that Malleable Prompting's version control and attribution features helped them more effectively evaluate outputs (8/12). For example, P11 emphasized that the the version control feature helped them *"break the linear interface and reuse previous prompts."* P01 described that the attribution feature enabled them to evaluate trade-offs in tasks that require careful inspection, *"I found two parts of my prompt were actually in a trade-off (emphasizing contributions vs. factual accuracy),"* and the participant utilized the attribution feature to *"guide"* them to think about *"which parts have impact on the results"* and *"which parts to change."* However, some participants preferred coarser granularity in attribution (3/12) over word-level visualization of influence. As P06 described: *"I judge based on the overall text whether it matches the tone I want. I wouldn't judge through a single word."* Two participants suggested a feature that allows the system to automatically suggest widgets that are inferred "in reverse" from a selection within output: *"select text here and have it tell me which widget can adjust that"* (P12).

How Users Map Widgets to Preferences: Participants reflected on their mental models of how to use Malleable Prompting and different strategies for adopting widgets. Sliders were preferred (11/12) for subjective and adjective-based attributes that vary by degree or ordinality such as tone, formality, warmth, and humor. Dropdowns mapped naturally to discrete categorical attributes (10/12) and surfaced hidden possibilities (8/12), as P07 reflected, *"... it expanded the options to, for example, junior or more senior IS (Information Systems) scholars. These choice ranges, before it expanded them, I myself hadn't thought about the possibility of further adjustment."* P04 found toggles *"good for objective things like [whether to use] paragraphs or actionable insights (bullet points)"*, also echoed by 6 other participants of being helpful for controlling whether to include bullet points, tables, or sections. However, 4 participants mentioned that in some cases, the toggling effect can be more easily achieved through the removal of certain expressions from the prompt.

Pros and Cons of Malleable Prompting: Although widgets were generally preferred by participants, some participants also identified specific scenarios where either NL prompting alone or Malleable Prompting offered more value. Prompting alone was often preferred for simple, well-defined tasks (8/12), where parametric control may introduce unnecessary cognitive overhead. Conversely, Malleable Prompting was preferred for style-driven writing (11/12), tasks that can benefit from reusable templates (7/12), and tasks with many competing and/or undefined requirements (5/12). For example, P10 identified that scenarios where content is fixed, but tone, formality, and register require iteration, would benefit most from our method. In addition, participants noted that their perception of the adjustment over sliders required calibration, especially for those attributes whose extremes are hard to imagine. P02 described that for some complex attributes (e.g., professional) they needed to *"test the maximum first to find where the boundary is,"* before intermediate values became meaningful.

7 Discussion

Our findings provide empirical insights about how users approach Malleable Prompting. Here, we discuss these insights and the implications around the two gulfs that motivated our work.

7.1 Bridging the Gulf of Execution

When and Why Malleable Prompting Helps: Malleable Prompting helped mitigate users' challenges in specifying and editing preferences in natural language [51]. We also found that users developed emerging mental models of when to use which type of inline widgets. For example, sliders were found to be an effective design that allowed users to modulate existing ordinal attributes. However, when users needed specific attributes that had not yet been surfaced, they often resorted to inserting or appending natural language prompts. This suggests that while widget controls benefit the adjustment of preferences, natural language should still be offered as a fallback mechanism for user-defined attributes. Future systems should also consider allowing users to easily transit between these two control modalities.

Adaptive Widget Complexity for Different Task Types: In terms of task type, users preferred NL prompting alone for simple, well-defined tasks, whereas Malleable Prompting was preferred for style-driven writing and exploratory tasks involving competing or underdefined requirements. This was likely because the parametric control offered by our approach allowed them to manipulate specific dimensions of the output without the ambiguity or trial-and-error inherent in purely linguistic interfaces. At the same time, such control could also introduce overhead for simpler tasks. Similar to the concept of matching UI complexity to task complexity [38], our findings call for future designs to incorporate mechanisms to surface widgets adaptively, such as triggering when users began work on exploratory tasks rather than always being offered.

Enrichment as Preference Discovery: Beyond adjusting existing attributes, participants also valued Malleable Prompting for discovering what attributes to control in the first place. Past research [40, 51] has suggested that it is easier for users to construct or "envision" [46] preferences through interactions. In Malleable Prompting, the action space of widgets (e.g., range of sliders, options in dropdowns) was viewed as inspiration to scaffold this construction because they offered concrete alternatives to help users identify preferences they did not know they had. To enhance this process, future work could explore additional implicit signals such as users' natural language edits to aid preference discovery and provide attribute suggestions on-the-fly.

7.2 Bridging the Gulf of Evaluation

From Linear to Modular Prompt Iteration: In standard chat interfaces, iterations are often organized chronologically, which makes it challenging to compare outputs or attribute differences to specific prompt changes [14, 56]. Our system restructures this by organizing iterations around changes in configuration of parameters rather than chat turns. This design was valued by participants, reflected through their preferences for the version control feature. Additionally, by making the mapping between the changes

in widget values and outputs explicit, Malleable Prompting turns evaluation from traditional holistic judgment to a more structured and comparative approach [36]. We also found that participants preferred coarser granularity in attribution over word-level visualization of influence. Attribution granularity can be designed as attribute-type-aware, such as using sentence-level attribution for tone and style or word-level attribution for localized constraints.

Calibrating Perception Through Boundary Probing: Our findings also suggest that users naturally adopt a calibration strategy enabled by continuous controls. P02's mention of *"test the maximum first"* reveals that continuous controls require aligning their perception of the slider's range before intermediate values become meaningful. This echos the challenge of "gulf of envisioning" [46] when interacting with LLMs. Future designs can consider showing example outputs at sliders' boundary values, or providing previews to help users align their perception of the slider's range.

7.3 Broader Applications

Here, we discuss the potential application of Malleable Prompting beyond the example system in §4 and use cases from our study.

Text Editor: Malleable Prompting enables prompt-side parametric control, which can be combined with output-side direct manipulation (e.g., DirectGPT [33]) into a text editor. One participant also explicitly mentioned that a "reverse widget" can be useful in the case where the user wants to indicate a specific output area where refinement is needed, and the system can thus infer and create an appropriate widget that only impacts the target range. Additionally, the range can also be user-defined through manual selection.

Prompt Library: Malleable Prompting also points toward reusable prompt templates with built-in controls. Several participants saw value in templates that could be shared across users, with personalization supported through user-specific slider calibrations, or reused across tasks with different attribute configurations. This direction connects to prior work on prompt reuse challenges [14] and composable prompt workspaces [1]. Designing and implementing such libraries would require a structured template schema that encodes attribute types, widget types, and valid ranges or options. Additional features such as automatic recommendation and adaptation can also be implemented to enhance re-usability of such templates for different use cases.

Agentic Workflow: The steering mechanism can potentially bootstrap beyond style-driven writing to broader agentic workflows. For example, controllable parameters could help steer code generation, data wrangling, information gathering, tool-use strategies, or brainstorming processes. In these settings, widgets may capture not only stylistic preferences but also behavioral ones, such as exploration depth, level of initiative, or degree of constraint in tool use. Supporting these applications would require validating and extending our preference taxonomy [25] for domains beyond content generation.

7.4 Limitations and Future Work

Our work has several limitations that could be addressed in future work. First, our study was conducted with a smaller user sample of 12 participants and findings might not generalize to users with broader background and used to work with other scenarios. Second, our analysis did not consider participants' expertise level in writing and prompt engineering, which may affect the participants' strategy and breakdowns when using Malleable Prompting. Third, our Jeopardy tasks focused on emails and explanations, while more open-ended creative tasks where preferences are less decomposable, such as poetry or fiction writing, should be considered.

Technically, our steering method provides an operational approximation for context-based control rather than globally grounded, fully disentangled semantic parameters. Subjective attributes such as "funny" and "professional" are context-dependent and not able to be fully disentangled. Accordingly, sliders should be interpreted as contextual control handles: although they do not represent universal semantic dimensions, they remain valuable for controlling preferences and steering models, as shown in our study. Similarly, our attribution method provides a local, post-hoc, probability-based signal intended as a perceptual handle rather than a precise causal explanation of the model's reasoning. Our technical evaluation should also be interpreted as a task- and model-specific validation for our system design rather than a definitive measure of semantic alignment or the causal faithfulness of this token-control method. Future work could explore better semantic disentanglement, causal explanations of widget effects, and more generalizable evaluations across models, settings, and tasks.

8 Conclusion

We introduce Malleable Prompting, an interactive prompting technique that expands preference expression from natural language prompts to in-line widgets, enabled by an LLM decoding method that steers generation and attributes each widget's influence on the output. Our user study (N=12) showed that Malleable Prompting helped users achieve target preferences more precisely and was perceived as more controllable and transparent than natural language prompting alone. We provide insights on how Malleable Prompting bridges the gulfs of execution and evaluation that exist in chat-based prompting, and offer design suggestions for how future systems can combine GUI widgets and NL prompting as an approach for more precise and transparent human-AI collaboration.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2119589. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This project was made possible in part by the Institute of Museum and Library Services RE-252329-OLS-22. Additionally, cloud computing resources for this project were obtained through CloudBank [37], which is supported by the National Science Foundation under Award No. 1925001. Literature discovery and research synthesis for this project were supported by Asta, a scientific AI assistant developed by the Allen Institute for Artificial Intelligence (Ai2).

References

[1] Rifat Mehreen Amin, Oliver Hans Kühle, Daniel Buschek, and Andreas Butz. 2025. Composable Prompting Workspaces for Creative Writing: Exploration

and Iteration Using Dynamic Widgets. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3706599.3720243

[2] Rifat Mehreen Amin, Oliver Hans Kühle, Daniel Buschek, and Andreas Butz. 2025. PromptCanvas: Composable Prompting Workspaces Using Dynamic Widgets for Exploration and Iteration in Creative Writing. arXiv:2506.03741 [cs] doi:10.48550/arXiv.2506.03741

[3] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3613904.3642016

[4] Tal August, https://orcid.org/0000-0001-6726-4009, View Profile, Kyle Lo, https://orcid.org/0000-0002-1804-2853, View Profile, Noah A. Smith, https://orcid.org/0000-0002-2310-6380, View Profile, Katharina Reinecke, https://orcid.org/0000-0001-7897-9325, and View Profile. 2024. Know Your Audience: The Benefits and Pitfalls of Generating Plain Language Summaries beyond the "General" Audience. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (ACM Conferences)*. Association for Computing Machinery, New York, NY, USA, 1–26. doi:10.1145/3613904.3642289

[5] Allan Bell. 1984. Language Style as Audience Design. *Language in Society* 13, 2 (June 1984), 145–204. doi:10.1017/S004740450001037X

[6] Lloyd F. Bitzer. 1968. The Rhetorical Situation. *Philosophy & Rhetoric* 1, 1 (1968), 1–14. jstor:40236733

[7] Jessica Y. Bo, Tianyu Xu, Ishan Chatterjee, Katrina Passarella-Ward, Achin Kulshrestha, and D. Shin. 2025. Steerable Chatbots: Personalizing LLMs with Preference-Based Activation Steering. arXiv:2505.04260 [cs] doi:10.48550/arXiv.2505.04260

[8] Stephen Brade, Bryan Wang, Mauricio Sousa, Sageev Oore, and Tovi Grossman. 2023. Promptify: Text-to-Image Generation through Interactive Prompt Exploration with Large Language Models. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3586183.3606725

[9] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative Research in Psychology* 3, 2 (Jan. 2006), 77–101. doi:10.1191/1478088706qp063oa

[10] Hugh Chen, Ian C. Covert, Scott M. Lundberg, and Su-In Lee. 2023. Algorithms to Estimate Shapley Value Feature Attributions. *Nature Machine Intelligence* 5, 6 (2023), 590–601.

[11] Yingchaojie Feng, Xingbo Wang, Kam Kwai Wong, Sijia Wang, Yuhong Lu, Minfeng Zhu, Baicheng Wang, and Wei Chen. 2024. PromptMagician: Interactive Prompt Engineering for Text-to-Image Creation. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (Jan. 2024), 295–305. doi:10.1109/TVCG.2023.3327168

[12] Raymond Fok, Joseph Chee Chang, Tal August, Amy X. Zhang, and Daniel S. Weld. 2024. Qlarify: Recursively Expandable Abstracts for Dynamic Information Retrieval over Scientific Papers. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3654777.3676397

[13] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. 2015. DataTone: Managing Ambiguity in Natural Language Interfaces for Data Visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. Association for Computing Machinery, New York, NY, USA, 489–500. doi:10.1145/2807442.2807478

[14] Katy Ilonka Gero, Chelse Swoopes, Ziwei Gu, Jonathan K. Kummerfeld, and Elena L. Glassman. 2024. Supporting Sensemaking of Large Language Model Outputs at Scale. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3613904.3642139

[15] Frederic Gmeiner, Nicolai Marquardt, Michael Bentley, Hugo Romat, Michel Pahud, David Brown, Asta Roseway, Nikolas Martelaro, Kenneth Holstein, Ken Hinckley, and Nathalie Riche. 2025. Intent Tagging: Exploring Micro-Prompting Interactions for Supporting Granular Human-GenAI Co-Creation Workflows. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–31. doi:10.1145/3706598.3713861

[16] Eliya Habba, Noam Dahan, Gili Lior, and Gabriel Stanovsky. 2025. PromptSuite: A Task-Agnostic Framework for Multi-Prompt Generation. arXiv:2507.14913 [cs] doi:10.48550/arXiv.2507.14913

[17] Jerry Zhi-Yang He, Sashrika Pandey, Mariah L. Schrum, and Anca Dragan. 2025. Context Steering: Controllable Personalization at Inference Time. arXiv:2405.01768 [cs] doi:10.48550/arXiv.2405.01768

[18] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685 [cs] doi:10.48550/arXiv.2106.09685

[19] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. 1985. Direct Manipulation Interfaces. *Hum.-Comput. Interact.* 1, 4 (Dec. 1985), 311–338. doi:10.1207/s15327051hci0104_2

[20] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. 1986. Direct Manipulation Interfaces. In *User Centered System Design*. CRC Press, Boca Raton, FL, 87–124.

[21] Rahul Jain, Amit Goel, Koichiro Niinuma, and Aakar Gupta. 2025. AdaptiveSliders: User-Aligned Semantic Slider-Based Editing of Text-to-Image Model Output. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. ACM, Yokohama Japan, 1–27. doi:10.1145/3706598.3714292

[22] Tae Soo Kim, Yoonjoo Lee, Yoonah Park, Jiho Kim, Young-Ho Kim, and Juho Kim. 2025. CUPID: Evaluating Personalized and Contextualized Alignment of LLMs from Interactions. arXiv:2508.01674 [cs] doi:10.48550/arXiv.2508.01674

[23] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2024. EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3613904.3642216

[24] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. LLMs Get Lost In Multi-Turn Conversation. arXiv:2505.06120 [cs] doi:10.48550/arXiv.2505.06120

[25] Seongyun Lee, Sue Hyun Park, Seungone Kim, and Minjoon Seo. 2024. Aligning to Thousands of Preferences via System Message Generalization. arXiv:2405.17977 [cs] doi:10.48550/arXiv.2405.17977

[26] Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting Human Preferences with Language Models. arXiv:2310.11589 [cs]

[27] Zhipeng Li, Yi-Chi Liao, and Christian Holz. 2026. Preference-Guided Prompt Optimization for Text-to-Image Generation. arXiv:2602.13131 [cs] doi:10.48550/arXiv.2602.13131

[28] Jiaju Ma, Lei Shi, Kenneth Aleksander Robertsen, and Peggy Chi. 2025. AmbigChat: Interactive Hierarchical Clarification for Ambiguous Open-Domain Question Answering. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3746059.3747686

[29] Stephen MacNeil, Andrew Tran, Joanne Kim, Ziheng Huang, Seth Bernstein, and Dan Mogil. 2023. Prompt Middleware: Mapping Prompts for Large Language Models to UI Affordances. arXiv:2307.01142 [cs] doi:10.48550/arXiv.2307.01142

[30] Chaitanya Malaviya, Joseph Chee Chang, Dan Roth, Mohit Iyyer, Mark Yatskar, and Kyle Lo. 2025. Contextualized Evaluations: Judging Language Model Responses to Underspecified Queries. *Transactions of the Association for Computational Linguistics* 13 (July 2025), 878–900. doi:10.1162/TACL.a.24

[31] V. K. Chaithanya Manam, Joseph Divyan Thomas, and Alexander J. Quinn. 2022. TaskLint: Automated Detection of Ambiguities in Task Instructions. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 10 (Oct. 2022), 160–172. doi:10.1609/hcomp.v10i1.21996

[32] Damien Masson, Young-Ho Kim, and Fanny Chevalier. 2025. Textoshop: Interactions Inspired by Drawing Software to Facilitate Text Editing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–14. doi:10.1145/3706598.3713862

[33] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. 2024. DirectGPT: A Direct Manipulation Interface to Interact with Large Language Models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3613904.3642462

[34] Aditi Mishra, Bretho Danzy, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2025. PromptAid: Visual Prompt Exploration, Perturbation, Testing and Iteration for Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31, 10 (Oct. 2025), 6946–6962. doi:10.1109/TVCG.2025.3535332

[35] Sheshera Mysore, Debarati Das, Hancheng Cao, and Bahareh Sarrafzadeh. 2025. Prototypical Human-AI Collaboration Behaviors from LLM-Assisted Writing in the Wild. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 16819–16846. doi:10.18653/v1/2025.emnlp-main.852

[36] Donald A. Norman. 1986. Cognitive Engineering. In *User Centered System Design*. CRC Press, Boca Raton, FL, 31–62.

[37] Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, et al. 2021. Cloudbank: Managed services to simplify cloud access for computer science research and education. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions*. 1–4.

[38] Nathalie Riche, Anna Offenwanger, Frederic Gmeiner, David Brown, Hugo Romat, Michel Pahud, Nicolai Marquardt, Kori Inkpen, and Ken Hinckley. 2025. AI-Instruments: Embodying Prompts as Instruments to Abstract & Reflect Graphical Interface Commands as General-Purpose Tools. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3706598.3714259

[39] Ben Shneiderman. 1981. Direct Manipulation: A Step beyond Programming Languages. In *Proceedings of the Joint Conference on Easier and More Productive*

Use of Computer Systems. (Part - II): Human Interface and the User Interface - Volume 1981 (CHI '81). Association for Computing Machinery, New York, NY, USA, 143. doi:10.1145/800276.810991

[40] Paul Slovic. 1995. The Construction of Preference. *American Psychologist* 50, 5 (1995), 364–371. doi:10.1037/0003-066X.50.5.364

[41] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M. Drucker, and Ken Hinckley. 2020. InChorus: Designing Consistent Multimodal Interactions for Data Visualization on Tablet Devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376782

[42] Arjun Srinivasan and John Stasko. 2018. Orko: Facilitating Multimodal Interaction for Visual Exploration and Analysis of Networks. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (Jan. 2018), 511–521. doi:10.1109/TVCG.2017.2745219

[43] Arjun Srinivasan and John Stasko. 2020. How to Ask What to Say?: Strategies for Evaluating Natural Language Interfaces for Data Visualization. *IEEE Computer Graphics and Applications* 40, 4 (July 2020), 96–103. doi:10.1109/MCG.2020.2986902

[44] S. S. Stevens. 1946. On the Theory of Scales of Measurement. *Science* 103, 2684 (June 1946), 677–680. doi:10.1126/science.103.2684.677

[45] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. 2022. Interactive and Visual Prompt Engineering for Ad-Hoc Task Adaptation with Large Language Models. arXiv:2208.07852 [cs] doi:10.48550/arXiv.2208.07852

[46] Hari Subramonyam, Roy Pea, Christopher Pondoc, Maneesh Agrawala, and Colleen Seifert. 2024. Bridging the Gulf of Envisioning: Cognitive Challenges in Prompt Based Interactions with LLMs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–19. doi:10.1145/3613904.3642754

[47] Mukund Sundararajan and Amir Najmi. 2020. The Many Shapley Values for Model Explanation. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Vienna, Austria, 9269–9278.

[48] Zhijie Wang, Yuheng Huang, Da Song, Lei Ma, and Tianyi Zhang. 2024. PromptCharm: Text-to-Image Generation through Multi-Modal Prompting and Refinement. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3613904.3642803

[49] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. ACM, New Orleans LA USA, 1–10. doi:10.1145/3491101.3519729

[50] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–22. doi:10.1145/3491102.3517582

[51] 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)*. Association for Computing Machinery, New York, NY, USA, 1–21. doi:10.1145/3544548.3581388

[52] Chao Zhang, Shunan Guo, Abe Davis, and Eunyee Koh. 2026. Narrix: Remixing Narrative Strategies from Examples for Story Writing. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. ACM, Barcelona Spain, 1–24. doi:10.1145/3772318.3790813

[53] Chao Zhang, Kexin Ju, Zhuolun Han, Yu-Chun Grace Yen, and Jeffrey M. Rzeszotarski. 2025. Synthia: Visually Interpreting and Synthesizing Feedback for Writing Revision. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3746059.3747703

[54] Chao Zhang, Xuechen Liu, Katherine Ziska, Soobin Jeon, Chi-Lin Yu, and Ying Xu. 2024. Mathemyths: Leveraging Large Language Models to Teach Mathematical Language through Child-AI Co-Creative Storytelling. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–23. doi:10.1145/3613904.3642647

[55] Chao Zhang, Kexin Phyllis Ju, Xinyi Lu, Yu-Chun Grace Yen, and Jeffrey M. Rzeszotarski. 2026. EvaluAId: Human-AI Collaborative Evaluation of Open-Ended Student Essays. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*. ACM, Barcelona Spain, 1–20. doi:10.1145/3772318.3790814

[56] Chao Zhang, Shengqi Zhu, Xinyu Yang, Yu-Chia Tseng, Shenrong Jiang, and Jeffrey M. Rzeszotarski. 2025. Navigating the Fog: How University Students Recalibrate Sensemaking Practices to Address Plausible Falsehoods in LLM Outputs. In *Proceedings of the 7th ACM Conference on Conversational User Interfaces (CUI '25)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3719160.3736618

[57] Zheng Zhang, Jie Gao, Ranjodh Singh Dhaliwal, and Toby Jia-Jun Li. 2023. VISAR: A Human-AI Argumentative Writing Assistant with Visual Programming and Rapid Draft Prototyping. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–30. doi:10.1145/3586183.3606800

[58] Jianqun Zhou, Yuanlei Zheng, Wei Chen, Qianqian Zheng, Hui Su, Wei Zhang, Rui Meng, and Xiaoyu Shen. 2025. *Beyond Content Relevance: Evaluating Instruction Following in Retrieval Models*. arXiv:2410.23841 [cs] doi:10.48550/arXiv.2410.23841

[59] Shengqi Zhu, Jeffrey Rzeszotarski, and David Mimno. 2026. Show or Tell? Modeling the Evolution of Request-Making in Human-LLM Conversations. In *Findings of the Association for Computational Linguistics: EACL 2026*, Vera Demberg, Kentaro Inui, and Lluís Marquez (Eds.). Association for Computational Linguistics, Rabat, Morocco, 5023–5034. doi:10.18653/v1/2026.findings-eacl.265

A Computation Overhead Analysis

We analyze the additional computation introduced by the two technical components of Malleable Prompting relative to standard single-pass decoding, which requires exactly N forward passes to generate an output of N tokens.

Prompt Modularization. At each decoding step i, computing the modulated distribution (Equation 3) requires evaluating $F_{a,P_0}(x_i)$ for each $a \in \mathcal{A}_{\text{mod}}$, including binary attributes in $\mathcal{A}^{\text{bin}}$ and continuous attributes in $\mathcal{A}^{\text{cont}}$. This process involves two forward passes per attribute: one with attribute a present, one without. In addition, one forward pass is required for the substituted base $p(x_i \mid x_{<i}, P_{\text{sub}})$.

However, the pass for $p(x_i \mid x_{<i}, \emptyset, P_0)$ is shared across all attributes in $\mathcal{A}_{\text{mod}}$, so the total per-step cost is $|\mathcal{A}_{\text{mod}}| + 2$ forward passes: one for P_{sub}, one for $\emptyset$, and one for each $a \in \mathcal{A}_{\text{mod}}$. Over N decoding steps, prompt modularization requires:

$$C_{\text{mod}} = (|\mathcal{A}_{\text{mod}}| + 2) \cdot N \text{ forward passes.} \tag{6}$$

Prompt Attribution. Computing the attribution $\phi_a(x_i)$ (Equation 4) requires evaluating $p_{\lambda_{\setminus a}}$, the modulated distribution with attribute a's weight set to zero. Since the logits from every context pass are already in memory from the previous prompt modularization, this leave-one-out distribution is obtained by arithmetically recombining existing logits, with no additional forward passes. Attributes in $\mathcal{A}_{\text{sub}}$ are handled by prompting an LLM to locate verbatim output spans, incurring one API call per attribute in $\mathcal{A}_{\text{sub}}$.

Total Overhead. The total number of forward passes required for one generation-plus-attribution cycle is:

$$C_{\text{total}} = C_{\text{mod}} + C_{\text{attr}} = (|\mathcal{A}_{\text{mod}}| + 2) \cdot N. \tag{7}$$

The overhead relative to standard single-pass decoding is therefore a factor of $(|\mathcal{A}_{\text{mod}}| + 2)$. To mitigate the practical wall-clock cost, we incorporate two acceleration techniques:

a) **Batched inference**: We process all $|\mathcal{A}_{\text{mod}}| + 2$ sequences in a single batched forward call per step, making full use of GPU parallelism so the factor-of-$(|\mathcal{A}_{\text{mod}}|+2)$ overhead does not multiply latency directly.

b) **Incremental KV-cache decoding**: To exploit the fact that all sequences append the same steered token at each step, their KV caches therefore remain synchronized throughout generation, so the model only needs to process one new token per sequence per step rather than reattending to the full context.

B Prompts

This section presents the prompt we used to instruct the LLM to enrich, parse, and segment prompts; generate options; and locate strings in the output for discrete attributes.

B.1 Enriching Prompts

System Prompt:

```
You enrich a given content creation prompt by inferring missing
user preferences.

Rules:
- Infer at most three missing preferences that the user did NOT
state but are relevant and useful for the given task in the
prompt.
- Use the below common preferences as a guide, but do not limit
yourself to just these.
- Integrate the inferred preferences naturally into the original
prompt.
- Ensure the enriched prompt remains clear and coherent.

Preference seed types (examples, not exhaustive):
- Formality: formal, informal
- Clarity: simple language, complex language
- Conciseness: concise, verbose/lengthy, clear, non-repetitive
- Vividness: use rhetorical devices (metaphors, personification,
similes, hyperboles, irony, parallelism)
- Format: breadth-first, depth-first, step-by-step, consistency,
deductive, inductive, parallelism, bullet points, narrative,
satisfy constraints, interactive, support stances
- Tone: agreeable, sympathetic, cooperative, modest, altruistic,
appreciative, forgiving, generous, kind, friendly, polite,
funny, gregariousness, assertiveness, friendliness,
extraversion, excitement-seeking, activity-level, cheerfulness,
energetic, enthusiastic, talkable, anxiety,
introspection/private self-consciousness, neuroticism, aloof,
anger, depression, immoderation, vulnerability, self-pitying,
self-beneficial, tense, touchy, unstable, worrying,
emotionality, intellect, adventurousness, intellectual openness,
liberalism, openness to experience, curious, authoritative,
persuasive, professional, playful
- Depth: general topic, specific topic, nuanced insights
- Creativity: artistic, insightful, original, imaginative, novel,
explorative creativity
- Efficiency: efficient, achievement-striving, self-discipline,
self-contained, contain rich info
- Practicality: practical, use supporting materials, tailored
examples and anecdotes, empowering actionable insights
- Audience: basic, novice, intermediate, advanced, expert,
general public, for my boss, for a peer collaborator, for a
family member, for a customer
```

Context Prompt:

Base prompt: `<insert the prompt>`

```
Infer missing preferences and return only the enriched prompt.
```

B.2 Parsing Prompts

System Prompt:
You extract controllable preferences from a given prompt for content creation.

Preference Types
1. Binary (on/off toggles)
For features that can be enabled/disabled.
- "Include examples"
- "Using bullet points"
2. Continuous (scalar intensity)
Use continuous for preferences that can vary along a spectrum (degree/strength/level). These are often adjectives or adverbs that describe how something should be (e.g., degree, mood, quality, price).
- "formal", "humorous", "casual", "friendly", "professional"
- "concise", "brief", "thorough", "verbose", "beginner-friendly"
- "cheap", "budget", "expensive", "luxury", "premium"
3. Categorical (discrete choices)
Use for mutually exclusive selections from a known/implicit set. These are often nouns or labels (audience, domain, locale, evaluation criteria, etc.).
- "for my boss"
- "in Manhattan, NY"
- "Chinese cuisine"
- "compare price and hardware"
- "explain like I'm five"
4. Numeric (quantities)
For counts and durations.
- "one-week extension"
- "3 paragraphs"

Output Fields
- id: lowercase-hyphenated identifier
- type: "binary" | "continuous" | "categorical" | "numeric"
- anchorText: EXACT substring from input (verbatim match required)
Preference extraction rules
- Returning an empty list is valid if no clear preferences are present.
- Do not invent preferences or infer beyond the text.
- Do not treat content type/genre (e.g., "blog post", "tweet", "email") as a preference.

Context Prompt:
Analyze this prompt and extract all controllable attributes:
`<insert the prompt>`

B.3 Segmenting Prompts

System Prompt:
You are a prompt segmentation assistant that separates a prompt into its core task and context prompts.

Given an original prompt and lists of attributes, you must:
1. Extract the BASE PROMPT: Remove all anchorText related to the attributes from the original prompt, keeping only the core task. The base prompt should be a neutral instruction without any context or binary prompt phrases.
2. Generate CONTEXT PROMPTS: For each attribute that needs to be turned into a context prompt, turn the provided anchorText into a standalone sentence. Preserve the attribute id.

Important rules:
- The base prompt should still be a complete, grammatical sentence
- Context prompts should start with "Be", "Keep", "Use", or similar directive

Example:
Original: "Write a concise and formal email to my boss asking for a one-week extension in two paragraphs, but keep the closing friendly. No bullet point."
Attributes to extract as context prompts:
[{ "id": "concise", "anchorText": "concise" },
{ "id": "formal", "anchorText": "formal" },
{ "id": "friendly", "anchorText": "friendly" }]

Output:
- basePrompt: "Write an email to my boss asking for a one-week extension."
- contextPrompts:
[{ "id": "concise", "prompt": "Be concise" },
{ "id": "formal", "prompt": "Be formal" },
{ "id": "friendly", "prompt": "Keep the closing friendly" }]

Context Prompt:
Original prompt: `<insert the prompt>`
Attributes to extract as context prompts:
`<insert the attributes and their anchored text>`
Segment this prompt into a base prompt and context prompts.

B.4 Generating Options

System Prompt:
You create categorical options for a selected phrase in a user prompt.

Rules:
- Provide 3-5 mutually exclusive options.
- "value" is a short label (1-3 words).
- "text" is the full phrase that could replace the anchorText in the original prompt.
- Always include the selected text verbatim as one of the options.
- Keep each option aligned to the original prompt context.
- NEVER generate synonyms or near-synonyms of the selected text. Each option must lead the prompt in a meaningfully different direction.
- Think about what ROLE the selected text plays in the prompt, then offer diverse alternatives that fit that same role but change the prompt's focus.

Context Prompt:
Prompt: `<insert the prompt>`
Selected text: `<insert the anchored text>`
Generate categorical options that replace the selected text.

B.5 Locating Strings

System Prompt:
`You are given a pair of prompt and output text. Your task is to locate exact, verbatim spans in the output text that are determined by the given portion in the prompt.`

`Rules:`
`- Only return substrings that appear exactly in the text.`
`- If there is no relevant span, return an empty list for that portion.`
`- Do not paraphrase or infer missing text.`

Context Prompt:
`Prompt: <insert the prompt>`

`Output text: <insert the output text>`

`Portion from the prompt (locate the exact output spans determined by this portion): <insert the anchored text>`

C Example Outputs for Different λ Values

This section presents example outputs generated with different λ for continuous attributes. The examples in table 5 show how changing λ alters the degree to which a selected attribute shapes the generated text.

D Technical Evaluation of Contextual Steering

Methods: We randomly sampled 100 prompts from the $\texttt{WildChat-WC}_{\texttt{Wr}}$ dataset [35] and 35 style attributes from the preference corpus from Lee et al. [25]. We test the method under 30 steering settings: 10 single-attribute settings (k=1), 10 two-attribute combinations (k=2), and 10 three-attribute combinations (k=3), where k denotes the number of attributes controlled simultaneously.

For each setting, we scan through the intensity of every active attribute over a 7-point scale and generated outputs under two conditions: 1) In the prompt-based scaling baseline, the intensity was expressed as an explicit numeric instruction prepended to the prompt (e.g., "*Apply the following writing style settings: formality: 2/7*") [32], and outputs were generated with standard decoding; 2) In our method, each intensity level $s \in \{1, \ldots, 7\}$ was mapped linearly to a steering coefficient spanning $\lambda \in [-10, 10]$, with the midpoint (s=4) recovers the unsteered distribution (λ=0), lower levels suppress the attribute, and higher levels amplify it. Both conditions used the same underlying model (`Qwen3-8B`) and decoding parameters (*temperature*=1, *top-p*=0), yielding 4,200 paired observations. An LLM judge rated the perceived intensity of each active attribute in each output on the same 1-7 scale; the judging protocol was validated against human ratings as described in §5.1.

Results: We report results from the `GPT-5.4` judge that our method achieving a steeper intensity-score linear regression slope on 29/35 dimensions ($p < .001^{***}$). More specifically, across all 35 dimensions, our method roughly doubles the steering effect over the prompt-only baseline ($\beta = 0.576$, 95% CI $[0.546, 0.607]$ vs. $\beta = 0.306$, 95% CI $[0.278, 0.335]$). We were able to separate the effect with a significant interaction model (score $\sim s \times$ condition; $\Delta\beta = +0.270$, 95% CI $[0.228, 0.312]$, $p < .001^{***}$). As shown in Table 6, the better steering effect holds even at when multiple attributes are steered at once ($\Delta\beta = +0.29$, $+0.40$, and $+0.17$ for $k = 1$, 2, and 3, respectively; all conditions show $p < .001^{***}$). When examined by individual dimensions, our method yields a steeper steering intensity slope on 30 out of 35 dimensions. Adopting contextual steering enables stronger effect over dimensions that prompting has close-to-none effect, such as Tense ($\beta = 0.10 \rightarrow 0.99$), Immoderation ($0.27 \rightarrow 1.06$), Unstable ($0.09 \rightarrow 0.87$), Anger ($0.17 \rightarrow 0.81$), and Forgiving ($0.23 \rightarrow 0.84$). Conversely, there are 5 out of 35 dimensions our method were not able to steer more effectively than prompting alone: Agreeable ($\beta = 0.32 \rightarrow 0.04$), Friendly ($0.34 \rightarrow 0.10$), Generous ($0.31 \rightarrow -0.08$), Warmth ($0.40 \rightarrow 0.25$), and Active ($0.51 \rightarrow 0.48$).

E Faithfulness of Attribution Highlights

Methods: To validate that the attribution highlights described in §5 faithfully reflect a slider's effect, we evaluate two properties: faithfulness, whether a token's displayed label predicts the direction of its probability change when the corresponding slider moves, and specificity, whether it predicts change for the token's own attribute rather than a different one. We reuse the setup of §D, namely the same model (`Qwen3-8B`), decoding parameters (*temperature*=1, *top-p*=0), $\texttt{WildChat-WC}_{\texttt{Wr}}$ prompts [35], and 30 steering settings (10 each for k=1, 2, and 3) with 10 prompts per setting. We take intensity level 5 as the reference condition and label each token *positive* if the attribution ratio $\phi_a(x_i) > 1.1$, *negative* if $\phi_a(x_i) < 0.9$, and *neutral* otherwise, as in the interface.

To obtain the ground-truth direction of change, we fix the sequence generated at the reference level and re-score it under a control vector that changes *only* the target attribute's intensity to a higher (7) or lower (3 and 1) level, computing the per-token change in log-probability

$$\Delta_a(x_i) = \log p_{\boldsymbol{\lambda}'}(x_i \mid x_{<i}, \mathcal{A}) - \log p_{\boldsymbol{\lambda}}(x_i \mid x_{<i}, \mathcal{A}), \tag{8}$$

where $\boldsymbol{\lambda}'$ differs from $\boldsymbol{\lambda}$ only in the entry for a. Both terms use the full modulated distribution (Equation 3), so the comparison reflects the deployed decoder rather than raw logits. A prediction *agrees* when the sign of $\Delta_a(x_i)$ matches the token's label. We report the *sign-agreement rate* with 95% bootstrap confidence intervals over settings, using neutral-band tokens as a null control that should sit near chance.

Results: As summarized in Table 7, highlighted tokens agree with the observed direction 81.4% of the time versus 67.2% for neutral tokens, a 14.2 pp gap above the null. The signal is stronger for positive than negative labels (85.1% vs. 58.1%), consistent with the asymmetry of the leave-one-out ratio, and it degrades only mildly with more simultaneous attributes (83.2% at k=1 to 80.8% at k=3). It is also attribute-specific: agreement on a token labeled for attribute a is 81.0% when a's own slider moves but falls to 60.8% when a different attribute's slider moves, close to the null. These results should be read as token-level rather than sequence-level, since the metric scores probability shifts of the same reference tokens under teacher forcing rather than free re-generation. Overall, they indicate that although the highlights are post-hoc indicators rather than causal explanations, they give a faithful and attribute-specific signal of how a widget reshapes the output distribution.

Table 5: *Example effects of λ across continuous attributes.* Outputs produced from the same prompts while varying the strength of target attributes during decoding. The top example shows the attribute *enthusiastic* for prompt "*Write a short post announcing a campus sustainability event in an enthusiastic tone*," and the bottom example shows *playful* for prompt "*Explain what a large language model is and what it can/cannot do in a playful way.*" In both cases, larger λ values produce a stronger realization of the target attribute while preserving the underlying content. Highlighted spans indicate wording influenced by the target attribute.

Attribute	λ	Example output excerpt
enthusiastic	0	Join us for Campus Sustainability Day on April 22nd. The event will include workshops, activities, and opportunities to learn about eco-friendly practices and environmental responsibility. Attend to learn how you can support a more sustainable campus community.
	1	Join us for Campus Sustainability Day on April 22nd! The event will feature workshops, activities, and opportunities to learn about eco-friendly practices and environmental responsibility. Come be part of a more sustainable campus community.
	2	Get ready to make a difference at Campus Sustainability Day on April 22nd! Join us for engaging workshops, hands-on activities, and inspiring opportunities to learn about eco-friendly practices and environmental responsibility. Come celebrate and help build a more sustainable campus community!
	3	Get ready to celebrate, connect, and make a difference at Campus Sustainability Day on April 22nd! Join us for exciting workshops, hands-on activities, and inspiring opportunities to explore eco-friendly practices and environmental responsibility. Let's come together and show our love for a greener future by building a more sustainable campus community!
playful	0	A large language model is an advanced type of artificial intelligence that is trained on vast amounts of text data to understand and generate human-like text. It can perform tasks such as answering questions, writing stories, coding, and translating languages, but it cannot truly understand or possess consciousness. Additionally, it may sometimes produce incorrect or biased information, so its outputs should be critically evaluated.
	1	A large language model is like a super-smart digital brain that's been trained on tons of text to understand and generate human-like language—think of it as a chatbot with a PhD! It can write stories, answer questions, translate languages, and even help with coding, but it can't truly understand emotions or have real-world experiences like humans do. Basically, it's a helpful tool with a lot of knowledge, but it's still just a bunch of words and patterns—it doesn't have a soul, just a lot of data!
	2	Imagine a super-smart, never-sleeping librarian who knows everything… well, almost everything. A large language model is like that librarian, trained on tons of books, chats, and stories, so it can chat, write, explain things, and even help you brainstorm—but it can't actually read a book or see the world like you do. Basically, it's your friendly AI sidekick who's great at talking, but still learning how to feel the vibe of a conversation!
	3	Imagine a super-smart gossip squad that's been training on every conversation, book, movie, and meme ever made—it's like having a party animal who's also a genius! This is a Large Language Model (LLM), and it can chat, write stories, answer questions, and even help you brainstorm your next big idea. But don't expect it to fold your laundry or tell you what your cat is thinking—it's great at words, but still learning how to do life hacks!

Table 6: *Slider-responsiveness regression.* Least-squares slopes β of judge score on intensity level (1-7). β is measured in judge-score points per slider step.

Scope	β Baseline [95% CI]	β Ours [95% CI]	$\Delta\beta$ [95% CI]	p
Overall	0.306 [0.278, 0.335]	0.576 [0.546, 0.607]	+0.270 [0.228, 0.312]	$< .001^{***}$
k=1	0.450 [0.382, 0.518]	0.741 [0.673, 0.809]	+0.291 [0.195, 0.388]	$< .001^{***}$
k=2	0.306 [0.255, 0.357]	0.711 [0.661, 0.760]	+0.404 [0.333, 0.476]	$< .001^{***}$
k=3	0.259 [0.220, 0.298]	0.432 [0.388, 0.476]	+0.173 [0.115, 0.232]	$< .001^{***}$

Table 7: *Attribution faithfulness and specificity.* Directional sign-agreement between a token's displayed attribution label and the observed direction of its probability change under teacher forcing. Higher is more faithful. The neutral band (H1) and off-diagonal (H2) rows serve as controls that should sit near chance. n is the number of prompt-attribute settings aggregated.

Condition	Sign-agreement [95% CI]	n
H1: Faithfulness		
Highlighted	81.4% [81.0, 81.7]	598
Neutral (control)	67.2% [66.4, 68.0]	598
by label polarity		
Positive	85.1% [84.8, 85.4]	598
Negative	58.1% [57.0, 59.2]	580
by no. of active attributes		
k=1	83.2% [82.5, 84.0]	98
k=2	81.3% [80.8, 81.8]	200
k=3	80.8% [80.4, 81.2]	300
H2: Specificity		
Diagonal (own)	81.0% [80.7, 81.3]	500
Off-diagonal (other, control)	60.8% [60.4, 61.3]	800

F User Study

F.1 Participant Information

We recruited 12 participants (7 female, 5 male) aged 24–37 (M = 26.67, SD = 3.68) via social networks and word of mouth. All participants reported using LLM tools, such as ChatGPT, Claude, or Gemini, for writing and editing tasks on a daily basis. Regarding their typical weekly writing volume (e.g., emails, reports, and posts), two participants reported writing 1–3 hours per week, six reported 4–7 hours, two reported 8–14 hours, and two reported 15+ hours. Participants were generally experienced with prompting, with a mean self-rated expertise of 5.17 (SD = 0.58) on a 7-point Likert scale (1 = Not at all, 7 = Very experienced). In the free-form exploration, participants used the system for a diverse range of genres, including academic introductions, homework essays, slide scripts, argumentative essays, social media posts, blog posts, and creative writing such as poetry. Table 8 provides detailed participant information.

Table 8: *Participant information.* This summary presents participants' demographic information and the self-chosen writing task they explored in the free-form exploration task in our user study. *Writing volume in a typical week; **Prompting experience was self-reported on a 7-point scale, where 1 = Not at all experienced and 7 = Very experienced.

ID	Gender	Age	Writing*	Prompting**	Writing Type in Free-Form Exploration
P01	Male	29	1–3 hours	5	Research article
P02	Female	24	8–14 hours	5	Homework essay and slide script
P03	Female	25	15+ hours	5	Activity announcement
P04	Male	24	4–7 hours	6	Argumentative essay
P05	Female	26	4–7 hours	5	Social media post
P06	Female	27	4–7 hours	5	Email
P07	Male	37	15+ hours	4	Research article
P08	Female	27	4–7 hours	5	Email
P09	Female	24	4–7 hours	5	Blog post
P10	Male	25	1–3 hours	5	Poem
P11	Male	24	4–7 hours	6	Reflection after watching a musical
P12	Female	28	8–14 hours	6	Meeting announcement

F.2 Baseline Interface

The baseline resembles common conversational LLM interfaces such as ChatGPT. In the baseline, users relied on NL prompting alone to generate outputs through a turn-by-turn conversation. We used the same underlying model (`Qwen3-8B`) for output generation as in our system to ensure a fair comparison. Fig. 6 shows an example screenshot of the baseline interface.

F.3 Jeopardy Tasks

For Jeopardy evaluation, we designed two writing tasks: an email task and an explanation task (Table 9). These tasks reflect a common writing scenario in which we must tailor content to different recipients or audiences [5, 6]. For each task, we prepared three target versions that differed along multiple preference dimensions based on our taxonomy of the preference space (Table 2) and prior work [25], such as audience, tone, depth, and formatting, while preserving the same underlying intent and factual content. Table 9 summarizes the tasks and target versions.

F.4 Interview Questions

(1) Walk me through how you approached the task with each interface. What did you do first, and how did you decide what to change next? How did your experience differ
(2) How did you decide which parts of the prompt should be made into widgets? Why those?
(3) How well did this interface let you steer the output toward what you had in mind? Can you point to a moment when you felt in control, or not in control?
(4) How easy was it to express what you wanted (tone, structure, depth, audience)? What felt hard to express or required trial and error?
(5) Did the interface match the way you think about your preferences? Were any preferences missing, misrepresented, or hard to specify?
(6) Did you realize any new preferences while working? What triggered that realization?
(7) When you made a change on the widgets, how easy was it to tell what changed in the output and why it changed? Can you give an example?
(8) How did you compare different drafts or versions? What cues did you use to decide which version was better or closer to the target?
(9) Tell me about a time you felt stuck, confused, or went in circles. What did you do to recover?
(10) If you had this interface for your own writing, what tasks or scenarios would you use it for? What would you change to make it fit better?

F.5 Prompt Enrichment Analysis

Prompt enrichment is designed to address the cold-start problem in preference exploration. Because enrichment changes prompts and can influence generation, it is optional: users may edit or remove it. We analyzed how users accepted or rejected the suggested content in enriched prompts as a validation of our prompt enrichment method. Among 24 first prompts sent to the system, 22 were submitted after enrichment; these had a 93% word-level acceptance, with 17 used directly, 4 edited, and 1 fully rejected.

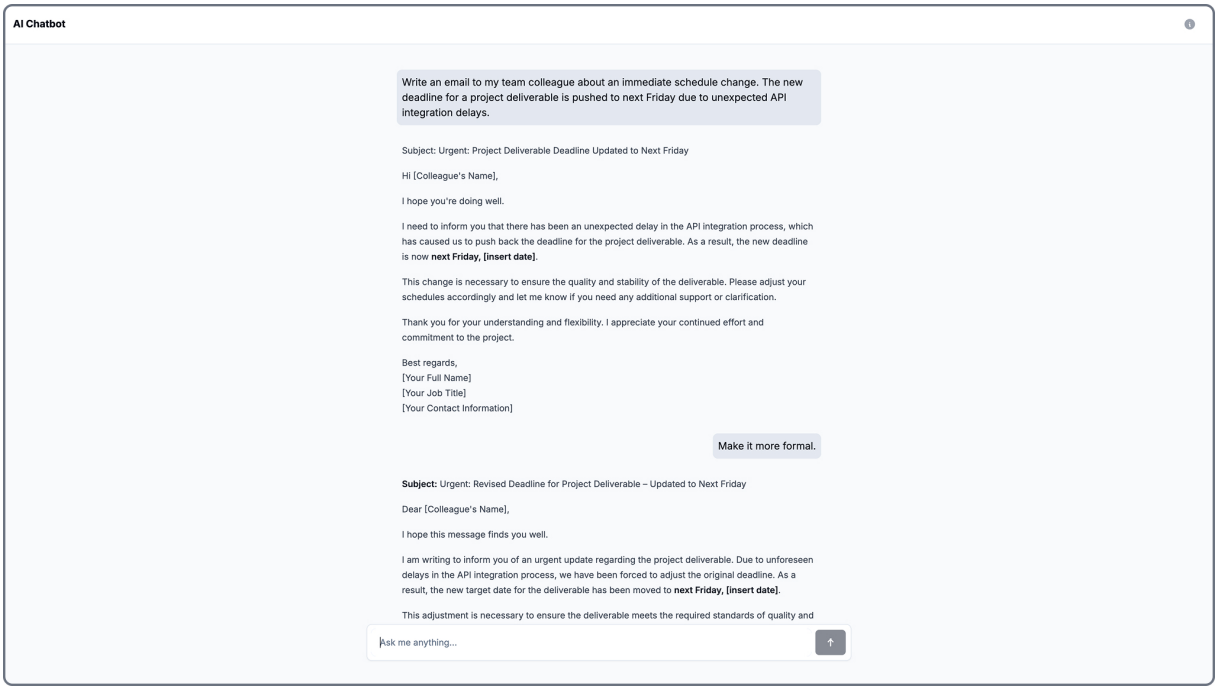

Figure 6: An example screenshot of the baseline interface

Table 9: *Jeopardy evaluation tasks and target preferences.* Each task includes three target versions that differ along preference dimensions (e.g., audience, tone, formality, format, conciseness) based on the taxonomy in Table 2 and prior work [25], while keeping task facts (e.g., dates and key details) constant across versions. Supplemental materials include all target outputs presented to participants.

Task	Scenario	Version 1	Version 2	Version 3
Email	Write an email about an immediate schedule change for a project deliverable, adapting it to different recipients.	**Recipient:** a team colleague **Formality:** informal **Tone:** informative **Format:** short paragraphs	**Recipient:** the manager **Formality:** formal **Tone:** persuasive **Format:** bullet points	**Recipient:** the client **Formality:** formal **Tone:** apologetic **Format:** short paragraphs
Explanation	Explain what a large language model is and what it can/cannot do, adapting the explanation to different audiences.	**Audience:** tech-savvy users **Tone:** professional **Conciseness:** detailed & technical **Format:** bullet points	**Audience:** general public **Tone:** approachable **Conciseness:** neutral **Format:** short paragraphs	**Audience:** middle-schoolers **Tone:** playful & encouraging **Conciseness:** simple & concise **Format:** short paragraphs