CoPrompter: User-Centric Evaluation of LLM Instruction Alignment for Improved Prompt Engineering

Ishika Joshi*†
ijoshi@adobe.com
Adobe Inc.
India

Simra Shahid*†
sshahid@adobe.com
Adobe Inc.
India

Shreeya Venneti*§
shreeya.manasvi@iiitb.ac.in
IIIT Bangalore
India

Manushree Vasu*§
mvasu6@gatech.edu
Georgia Institute of Technology
USA

Yantao Zheng
yantaoz@adobe.com
Adobe Inc.
USA

Yunyao Li
yunyaol@adobe.com
Adobe Inc.
USA

Balaji Krishnamurthy
kbalaji@adobe.com
Adobe Inc.
India

Gromit Yeuk-Yin Chan
ychan@adobe.com
Adobe Research
USA

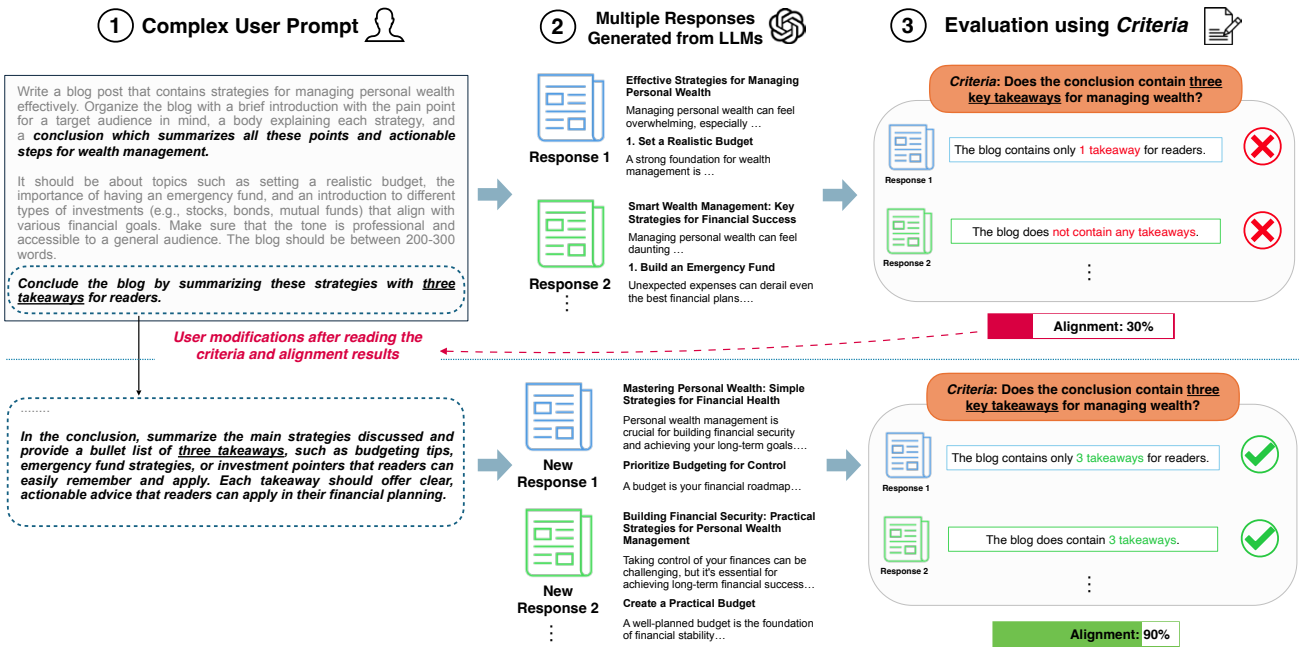

Figure 1: Overview of CoPrompter framework: Given a ① complex prompt from the user and ② multiple responses generated by LLMs, ③ CoPrompter generates a set of *criteria* to assess the responses and produce an alignment score. Users could iteratively refine and reflect on their prompts to result in a prompt that achieves better alignment.

ABSTRACT

Ensuring large language models' (LLMs) responses align with prompt instructions is crucial for application development. Based on our formative study with industry professionals, the alignment requires heavy human involvement and tedious trial-and-error especially when there are many instructions in the prompt. To address these challenges, we introduce CoPrompter, a framework that identifies misalignment based on assessing multiple LLM responses with criteria. It proposes a method to generate *evaluation criteria questions* derived directly from prompt requirements and an interface to turn these questions into a user-editable checklist. Our user study with industry prompt engineers shows that CoPrompter improves the ability to identify and refine instruction alignment with prompt requirements over traditional methods, helps them understand where and how frequently models fail to follow user's prompt requirements,

* First authorship. † Principal Investigators.
§ Work done in Internship with Adobe Media and Data Science Research.
Correspondence to Ishika Joshi (ijoshi@adobe.com), Simra Shahid (sshahid@adobe.com).

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and helps in clarifying their own requirements, giving them greater control over the response evaluation process. We also present the design lessons to underscore our system's potential to streamline the prompt engineering process.

CCS CONCEPTS

• **Human-centered computing → User interface toolkits**.

KEYWORDS

HCI, LLM Evaluation, Prompt Optimization

ACM Reference Format:
Ishika Joshi[*†], Simra Shahid[*†], Shreeya Venneti[*§], Manushree Vasu[*§], Yantao Zheng, Yunyao Li, Balaji Krishnamurthy, and Gromit Yeuk-Yin Chan. 2025. CoPrompter: User-Centric Evaluation of LLM Instruction Alignment for Improved Prompt Engineering. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 24 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large language models (LLMs) have become ubiquitous in a wide range of applications, finding use across billion-dollar industries for tasks ranging from basic operations to critical applications [5–7, 19, 33]. Prompting has emerged as the primary method for developers to interact with LLMs, enabling rich, Generative AI-driven experiences.

Given that many critical products and services rely on LLMs and prompts, ensuring that the generated contents align with user expectations is crucial.

Recent works have explored emerging themes of *misalignment* between prompt engineers and model responses. Drawing from the definition provided by Terry et al. [64], we define *misalignment* as "LLMs generating outcomes that deviate from the goals or expectations of the user, often leading to unintended or undesirable effects". Some misalignments that have been explored in literature include instruction overlooking and misinterpretation [18, 25, 53, 70, 75, 77] hallucinations [23, 26, 34, 42, 45], generation of harmful [16, 31, 49, 67, 69] or biased [11, 44, 66, 72, 74] contents, etc. This often stems from the inherent sensitivity of LLMs to the words within the prompt and the overall structure of prompts, where even minor changes can lead to drastically different outputs [39, 40, 43, 56].

Despite efforts being made towards improving the instruction-following ability of LLMs by training them on instruction datasets [21, 32, 38, 46], existing approaches often fall short of being consistently aligned with all the instructions in long-form prompts with numerous instructions (Figure 1). These prompts tend to experience greater misalignment not only due to the complexity of multiple instructions but also the variations in instruction writing styles that causes the model to overlook or misinterpret the prompt engineer's intended outcomes.

To better understand the practical challenges faced by prompt engineers, we conducted a formative survey with 28 professionals working across various domain-specific industry applications (Section 3). Based on the survey, the misalignment issues such as *overlooked instructions, inconsistent responses, instruction misinterpretations*, and *incorrect assumptions* are frequent, particularly when dealing with complex prompts containing five or more instructions. The current workflow for prompt engineering is thus *time-consuming* and *tedious*, often requiring over ten iterations and *manual inspection* of responses in a largely *trial-and-error* process.

In response to these challenges, we introduce CoPrompter, a novel tool that helps prompt engineers systematically identify and address areas of misalignment between multiple LLM outputs and their requirements. CoPrompter achieves this by first breaking down user requirements into atomic instructions, each transformed into criteria questions. With the criteria and LLM outputs, CoPrompter generates detailed misalignment reports at the instruction level. This granular approach allows users to quickly pinpoint where misalignments occur and identify which criteria fail most frequently, offering a systematic way to prioritize prompt refinements. As a *user-in-the-loop* tool, CoPrompter provides prompt engineers with control over the evaluation process, allowing them to customize criteria to better reflect specific requirements and adapt to evolving needs.

This work provides the following contributions:

- We introduce CoPrompter, a novel user-in-loop system that helps prompt engineers align complex prompts with their requirements through systematic evaluation and customizable criteria control.
- A technical pipeline that generates detailed misalignment reports, allowing prompt engineers to pinpoint specific areas in need of improvement and prioritize prompt adjustments.
- We present findings from a formative study with 28 prompt engineers, contributing to the understanding of the challenges in crafting complex prompts and the common misalignments observed with LLM responses.
- We conduct a comprehensive user evaluation with 8 industrial prompt engineers, demonstrating that CoPrompter effectively identifies misalignments, helps in prompt refinement, and adapts to evolving requirements, significantly improving upon traditional manual methods.
- A focused user evaluation with high System Usability Scale (SUS) scores further reflects user confidence in its seamless integration into prompt improvement workflows.

2 RELATED WORK

2.1 Challenges in Aligning LLM Outputs with Prompt Instructions

Among the various alignment goals [12, 60] between prompt instructions and LLM outputs, in this work we focus on understanding LLM's ability to follow prompts with multiple instructions [21, 22, 27, 38, 41, 48, 50, 68]. A model is considered *misaligned* when it deviates from expected outcomes, producing ambiguous or suboptimal responses [4, 17, 71], arising from overlooked instructions, hallucinated content, or inconsistent responses across interactions.

Prior studies suggest improving alignment through supervised fine-tuning (SFT), preference tuning (DPO [51], RLHF [63], ORPO [48]), and instruction tuning. Despite these efforts, challenges persist, with recent works reporting ongoing issues.

The HCI scholarship has explored these challenges and attributed them to discrepancies in Human AI communication [30]. While

alignment has been studied from a standard benchmark perspective, researchers in HCI communities have highlighted the dynamic nature of alignment. Shen et al. emphasized that alignment is a bi-directional concept, requiring both the alignment of AI systems to human factors and the adaptation of humans to the capabilities of AI [61, 76]. Moreover, unpredictability, lack of transparency, value misalignment, and inherent complexity in understanding response behaviors while interacting with LLMs have been identified as common causes of misalignment [13, 73, 73]. Addressing misalignments becomes a complex process for prompt engineers who often try to optimize prompts while navigating these challenges. This highlights the need to build systems that bridge the gap between LLMs and prompt engineers by easing the process of identifying misalignments systematically, assisting prompt engineers in discovering the cause of misalignment and addressing it for streamlined prompt improvement.

2.2 Systematic Assessment of LLM Outputs

Researchers have explored evaluation criteria and LLM-as-a-Judge approaches to systematically assess and improve LLM outputs. A criterion provides a structured measure to determine if an LLM response meets specific conditions, typically including guidelines, ideal and non-ideal examples, and a scoring rubric [36]. Previous work has applied criteria to assess coherence, naturalness, and grounding in summarization tasks [8, 36, 37].

Several tools have been developed to support prompt engineers in evaluating responses [2, 24, 29, 30, 57, 59], such as PromptsRoyale[1] and promptfoo[2]. ChainForge [2] comprises of functionalities such as prompt generation, testing across multiple LLMs, and side-by-side comparison. SPADE [57] generates criteria based on prompt changes to analyze response quality, while EvalLM [29] turns prompt design into a collaborative process with the LLM acting as an evaluation assistant. EvalGen [59] extends ChainForge, enabling automated criteria checks and report generation on alignment with user-defined requirements. Further, it allows users to assign ground truth to each criterion (as a binary yes or no) which they coin as candidate assertions, and generate reports on criteria alignment.

Despite these advancements, existing tools do not help prompt engineers systematically identify causes of overall misalignment between user expectations and generated responses, often requiring manual inspection and arbitrary adjustments. To address this gap, we expand on the human-in-the-loop evaluation framework by Shankar et al. [59], using criteria derived from atomic instructions within prompts to evaluate alignment.

We build upon the framework in three key areas: (i) Instead of Python assertions, we automatically create criteria derived from the entire user prompt input, allowing for more complex criteria that serve as a checklist of sub-instructions within the prompt. The criteria generated also inform users of the nature of these criteria for eg. identifying any subjectivity and can be interpreted differently. (ii) Next, we provide support for customizable criteria evaluation. This enables prompt engineers to evaluate their changing requirements at any point in the form of editable criteria questions. (iii) Finally, our work differs by emphasizing providing detailed reports of LLM response alignment to user specified guidelines/requirements, which helps users to iterate on their prompts more systematically.

[1] https://www.promptsroyale.com/
[2] https://www.promptfoo.dev/

3 FORMATIVE STUDY

The motivation for CoPrompter stems from a formative study involving a survey of 28 industry prompt engineers working on domain-specific applications including content generation, chatbots, etc. Participants were selected from a multinational IT company based on their experience in prompting for product development or research. The survey was distributed via internal mailing lists, ensuring targeted outreach. Identities remained confidential, and participants were informed about the study's purpose and data usage. Responses were collected using Likert scales and analyzed through open coding and Thematic Analysis. To better understand their prompt engineering process and the issues they encounter, we asked the participants detailed questions (Appendix 3). Specifically, we asked questions around three key areas:

3.1 Instruction Styles

We examined how prompt engineers structure their prompts, focusing on (i) number of instructions, (ii) categories of instructions such as context-setting, format specifications, and style or task-specific directives, and (iii) styles such as concise lists or detailed step-by-step guides. About **64% of participants typically include 5-10 instructions in their prompts, while over 55% use more than 10**. These prompts often include task-specific details, formatting rules, and additional context about tasks. Most participants prefer concise, point-based lists or examples to clarify instructions, while a few use paragraph-style prompts.

3.2 Challenges of Instruction Following Misalignment

We surveyed participants on discrepancies between LLM responses and prompt instructions, asking them to rate alignment on a 1-to-5 scale. Most rated initial alignment as 3 or 4, indicating that while responses are generally aligned with the prompt, they are not always precise. When asked about the effort required to achieve optimal responses, most participants reported needing 10+ attempts and evaluating 40+ responses.

Participants also reported some common misalignments like **ignored instructions, hallucinated content, misinterpreted details, and inconsistent responses**. Complex prompts with detailed task or formatting instructions were especially prone to misalignment, even when clearly stated in the prompt. Additional details are provided in Appendix 3.

3.3 Strategies for Navigating Instruction Following Misalignment

Participants used various strategies to correct misalignment, such as improving misinterpreted instructions by adding context and breaking down complex tasks into smaller steps. Reordering instructions for clarity was also a common approach. However, this **process was described as time-consuming and trial and error**. These challenges highlight the need for a more structured and systematic solution for prompt improvements.

4 DESIGN GOALS

Based on the challenges identified by our formative study and literature review, we formalize three design goals to guide the development of CoPrompter:

- **(DG1)**: **Align the complex prompt's requirements through systematic evaluation.** Participants relied on trial-and-error to correct overlooked or misinterpreted instructions in complex prompts without a structured verification method. CoPrompter addresses this by breaking guidelines into checklist-based evaluation criteria, converting atomic instructions into questions, and ensuring thorough assessment of user requirements. It also applies tailored evaluation methods for each criterion, streamlining the detection and resolution of misalignments.
- **(DG2)**:**Refine prompts with insights from misalignment reports between model responses and instructions.** CoPrompter generates detailed reports by analyzing alignment and misalignment at the instruction level. These reports highlight unmet requirements and frequently failing criteria, helping prompt engineers prioritize adjustments. By pinpointing problematic instructions, CoPrompter provides actionable insights to refine prompts and improve alignment with intended goals.
- **(DG3)**: **Support evolving requirements and continuous prompt refinement.** Prompt engineering is an iterative process, where user requirements may shift as prompt engineers gain new insights or refine their understanding of the task. CoPrompter enables continuous updates to evaluation criteria, allowing prompt engineers to add, delete, or modify questions as needed. It also highlights subjective criteria, prompting users to clarify them further. This user-in-the-loop approach ensures adaptability, keeping evaluation criteria aligned with evolving task requirements.

5 COPROMPTER: SYSTEM DESIGN

In this section, we describe how CoPrompter system meets the design goals (Section 4). Section 5.1 outlines the User Interaction Workflow (Figure 2). Section 5.2 covers the backend implementation details (Figure 3).

5.1 User Interaction Workflow

CoPrompter offers an iterative workflow where users define, refine, and evaluate prompt responses. As illustrated in Figure 2(zoomed-in images in the Appendix5), the interface enables users to modify evaluation criteria, generate prompt responses using LLMs, and assess alignment with their requirements.

5.1.1 ***Build Your Evaluator:*** In the 'Build Your Evaluator' stage (a,b,b1 in Figure 2, users set up the evaluation framework based on their specific task requirements. Users begin by providing prompt requirements or guidelines, outlining what they expect in generated responses. This can be a prompt draft or guidelines that include the instructions that the user wants to evaluate in the generated prompt responses. This stage is supported by two backend modules.

① **Evaluation Criteria Generation Module**. This module breaks down these high-level guidelines into detailed evaluation questions, known as criteria questions (CQs) which help in comparing the alignment between the generated prompt responses and user's intended requirements.

For example, as shown in Figure 3, we assume the user persona of a content creator who wants to use LLMs to generate finance blogs. The user may provide a guideline that checks if the blog's conclusion includes a '*bullet list of specific takeaways with actionable advice for investment options*'. This instruction in the guideline is decomposed into two *atomic* instructions: '*bullet list of three specific takeaways*' and '*actionable advice for investment options*'. These atomic instructions are then converted into criteria questions:

(1) **Extracted Atomic Instruction:** *The conclusion should include a bullet list of three specific takeaways.*
Corresponding Criteria Question: Does the conclusion include a bullet list of three specific takeaways?
Ground Truth: Yes

(2) **Extracted Atomic Instruction:** *The takeaways must cover actionable advice for choosing suitable investment options.*
Corresponding Criteria Question: Does the list of takeaways include actionable advice for choosing suitable investment options?
Ground Truth: Yes

For each criteria question we also generate ground truth labels, and priority tags (e.g., main task, subtask, or format-related) that help in structured evaluation. CoPrompter also provides metadata tags, such as content vs. style categorization and question subjectivity. This ground truth is crucial for computing alignment and is compared with LLM-generated ground truth for each prompt response. Initially, users may only have a general understanding of their task requirements, but they can refine their criteria as their needs evolve during the evaluation setup.

② **Update Criteria Module**: After generating initial criteria questions, users can refine them by editing, deleting, or adding new criteria. This user-in-the-loop module offers flexibility and control, allowing updates to questions, ground truth, or priority tags as evaluation goals become clearer. Figure 3 outlines examples of specific edits and other ways to refine criteria.

- Edit: Users can edit a criteria question to make it more specific, they can include definitions or examples as shown in Figure 3. Users can view categorical tags for each criterion to identify subjective areas needing clarification. The formative study revealed that subjective instructions often cause misalignment due to differing interpretations between the prompt engineer and the model. To address this, CoPrompter classifies criteria as either 'Subjective' or 'Objective,' helping users identify those that require additional context. Figure 2 highlights these tags in the UI (screen b).
- Delete: If the user feels that a criteria is not important for evaluation or is redundant by another criterion, users can choose to delete it.
- Add New Criteria: Users can add new criteria for an existing guideline instruction or a completely new requirement to capture additional requirements. The types of criteria that can be added can be found in Appendix 1.

The **Build Your Evaluator** stage offers a flexible framework for evaluating alignment by providing user-in-the-loop control over question content, ground truth, and tags. This approach allows users

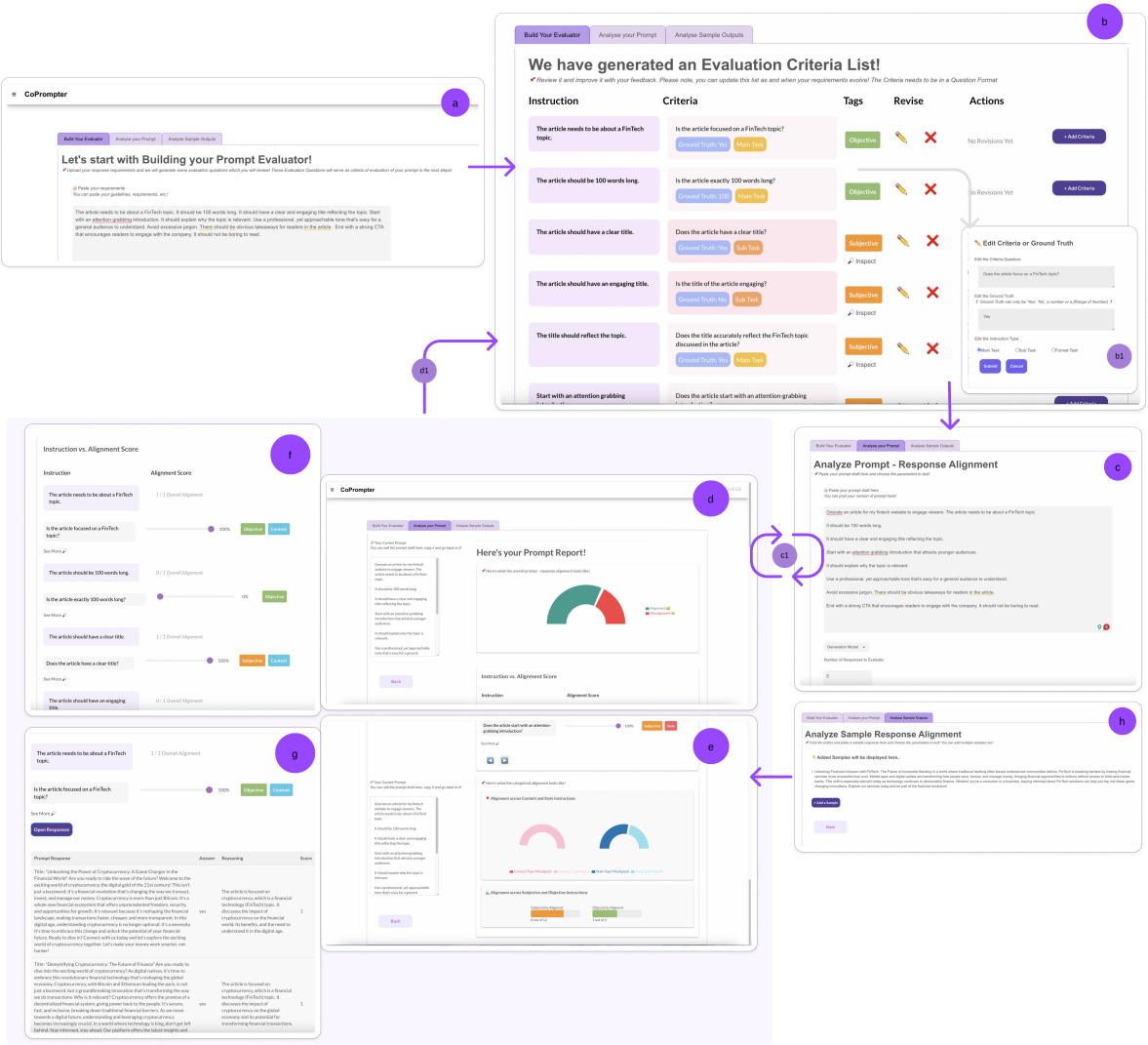

Figure 2: Users enter prompt requirements in the *Build Your Evaluator* tab, extract and modify atomic instructions and criteria questions, analyze prompts in the *Analyze Your Prompt* tab, evaluate responses based on criteria to generate alignment scores in the Alignment Report Card, view detailed scores and rationale, adjust prompts iteratively, and test sample responses against evaluation criteria with alignment categorized by content, style, and instruction type.

to continuously refine their criteria to match evolving task requirements. Users can then *save their criteria* and proceed to the next stage for evaluating their prompt.

5.1.2 ***Analyse Your Prompt:*** In this stage, users evaluate how well generated responses align with the criteria defined in the previous stage (c,c1,d,e,f,g in Figure 2). This stage is supported by two backend modules:

③ **Prompt Response Generation Module**. Within the 'Analyse Your Prompt' tab, users input their prompt draft, select a language model, and specify the number of responses they wish to generate. This module processes these inputs and generates multiple responses specified by the user using the LLM of their choice. These responses are independently evaluated against each saved criteria.

④ **Alignment Report Card Module**.

After the responses are generated, the Alignment Report Card Module evaluates each response against the saved criteria questions, providing a structured report on alignment that shows where responses meet or don't align with user requirements (criteria). This report is presented in an interactive **Prompt Alignment Report** that displays alignment scores in different dimensions: overall, by category, and for each specific criterion.

To start, the report shows a pie chart summarizing the number of aligned and misaligned criteria across all responses, giving users a quick sense of how well the responses match their expectations. Next,

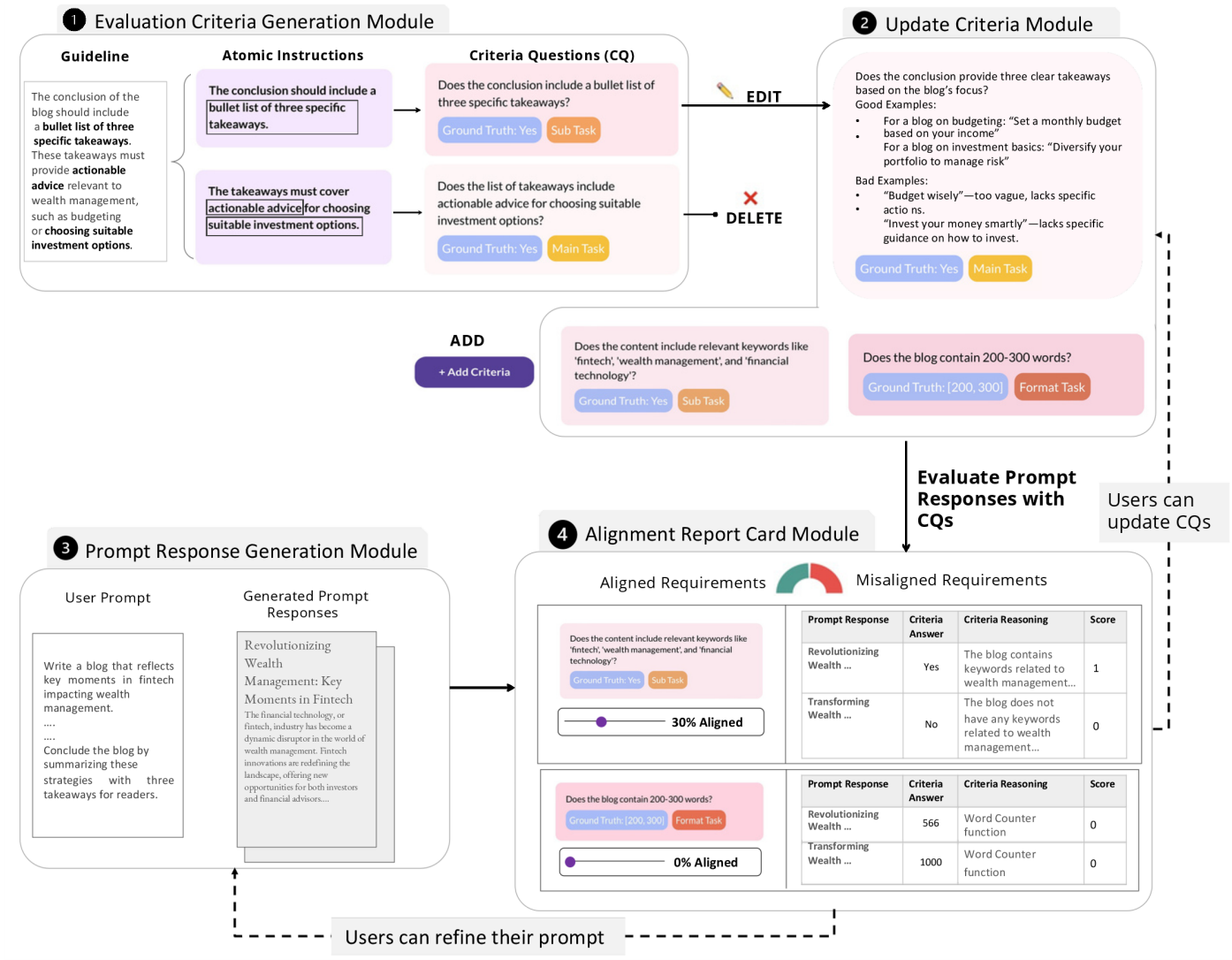

Figure 3: The CoPrompter system workflow begins with (1) the Evaluation Criteria Generation Module, where user prompt requirements (guidelines) are broken down into atomic instructions, which are then formulated as criteria questions (CQs) with ground truth labels and metadata tags (e.g., main task, subtask, or format-related). Users can adjust these criteria in (2) the Update Criteria Module by editing, deleting, or adding new CQs. Next, in (3) the Prompt Response Generation Module, users input their prompt, select an LLM, and generate responses. The responses are evaluated against the defined criteria, with results displayed in (4) the Alignment Report Card Module. This report shows aligned and misaligned requirements, allowing users to explore misalignments in detail. Based on the feedback, users can refine their prompt or criteria, iterating as needed to improve alignment.

users can explore a detailed breakdown for each Atomic Instruction and its matching Criteria Question. This breakdown shows exactly which requirements are being followed and where there are gaps. Users can click 'See More' to view each response, its alignment score, and an explanation from CoPrompter, providing transparency in how each response is scored.

For example, as shown in Figure 3, the report evaluates two user-defined criteria:

- Checking for Keywords: This criterion checks for the presence of keywords like 'wealth management' in the blog, with only 30% of generated blogs meeting this criterion. On the right side, a table displays results similar to the 'See More' option in the UI. In this example, the first blog includes the keywords, receiving a *yes* in the Criteria Answer column and a score of 1, indicating alignment with user-approved ground truth. The second blog lacks these keywords, resulting in a score of 0 due to a mismatch with ground truth.

- Word Count: This criterion ensures each blog adheres to a user-specified word count of 200-300 words. However, neither generated blog met this limit, with the first at 566 words and the second at 1000 words, resulting in a score of 0. The Criteria Reasoning indicates that a word counter function, rather than an LLM, was used to validate this criterion.

In addition to individual scores, the Prompt Alignment Report includes a Categorical Analysis that organizes scores by instruction types, like Content, Style, Subjective, and Objective. This helps users see which types of instructions may need improvement for better alignment.

After reviewing these scores, users can make edits directly in the Prompt Panel on the left side of the screen. They can adjust any misaligned instructions and rerun the prompt to get updated scores. If users identify new constraints or realize that their needs have changed, they can go back to the **Build Your Evaluator** tab to add or adjust criteria, ensuring the evaluation process stays aligned with their evolving needs.

This process lets users improve both their prompt and criteria based on feedback from CoPrompter, creating an iterative, user-friendly approach to refining their prompts for better results.

5.1.3 ***Analyse Sample Outputs:*** In addition to generating responses, users can test alignment by uploading their own sample outputs (for example, finance blogs) through the 'Analyse Sample Outputs' tab. This interface allows users to evaluate how well their examples align with the defined criteria. By testing real examples, users can confirm that the criteria effectively capture their requirements. Moreover, it can also be used to improve task understanding and requirements by updating criteria. After uploading sample responses, users are presented with a 'Sample Output Alignment Report Card', similar to the 'Prompt Alignment Report Card', which shows alignment scores for each uploaded example based on the saved criteria.

In the following sections, we provide the implementation details for each module.

5.2 Implementation

CoPrompter is built on a React front end which interacts with the backend through FastAPIs. The front end is designed with close consideration given to user convenience despite the complexity of the system. The use of information hierarchy, user guides, associative colors, and visual harmony is maintained for a good user experience. In this section, we describe the implementation details of each of the four primary modules of CoPrompter. We use three types of LLMs: the **User-Specified LLM** (LLM_{User}) for generating prompt responses based on user input, the **Criteria Generation LLM** (LLM_{CGen}) for generating atomic instructions, criteria questions, and metadata tags, and the **Evaluator LLM** (LLM_{Eval}) for evaluating responses against criteria. Our framework is designed to be architecture-agnostic and does not impose constraints on the underlying LLM used like GPT models, Llama models, Claude models, Gemini, etc. that fits their specific needs, ensuring broad applicability. In this work, we use GPT-4o with temperature 0 for LLM_{CGen} and LLM_{Eval}.

① Evaluation Criteria Generation Module

In this module, the user's input guidelines are transformed into specific evaluation criteria questions. The process involves the following steps:

Step 1: Input Guidelines. The user provides their prompt requirements or 'guidelines'. These guidelines include instructions that the user wants to ensure are followed in the generated prompt responses.

Step 2: Task Objective: To maintain context that might be lost during decomposition, the system also extracts a task objective. This task objective is a concise summary of the main goal of the user's task. It helps contextualize each atomic instruction when generating criteria questions. For example, the task objective for the prompt in Figure 3 is 'Generate a blog on wealth management.' The prompt for task objective is in the Appendix 8.

Step 3: Decomposing Guidelines into Atomic Instructions. The guidelines are decomposed into atomic instructions, each representing a single, clear requirement extracted from potentially compounded sentences. This ensures that each criterion evaluates only one aspect, avoiding overloaded criteria with multiple requirements and reducing complexity in the subsequent evaluation.The prompt for decomposing into atomic instructions is in the Appendix 8.

Step 4: Criteria Question Generation. For each atomic instruction, the system rephrases it as a question, which we refer to as a 'criteria question'. A criterion consists of:

- A **question** that checks if the generated prompt response meet its corresponding atomic instruction from the guidelines.
- The **ground truth** expected answer to the criteria question based on the user's guidelines. These can be either 'Yes'/'No', a specific number or a range (e.g. [200-300]). The motivation for binary score is to aggregate the overall criteria coverage for an overview purpose in CoPrompter as an interactive UI, which serves the users as a visual analytics perspective[20]. It is also a common alignment rating strategy for LLM evaluation systems [58, 59].
- The inferred **priority** from the guideline. We categorize a criterion into one of the three:

(1) 'Main Task', if the criterion corresponds to an instruction from the guideline which is fundamental to the task's core logic. These also include instructions with explicit emphasis (e.g., using phrases like 'make sure' or 'important').
(2) 'Sub Task', if the criterion corresponds to an instruction that is added to support the 'Main Task' including further details or context.
(3) 'Format Task', if the criterion corresponds to an instruction that are about formatting or style of the generated responses.

Examples of these priority tags, along with criteria based on a guideline, are shown in Figure 4.

To enable automatic evaluation, we categorize each criterion based on how it should be evaluated, guiding the LLM_{Eval} to select the appropriate method. Criteria are grouped into two main categories:

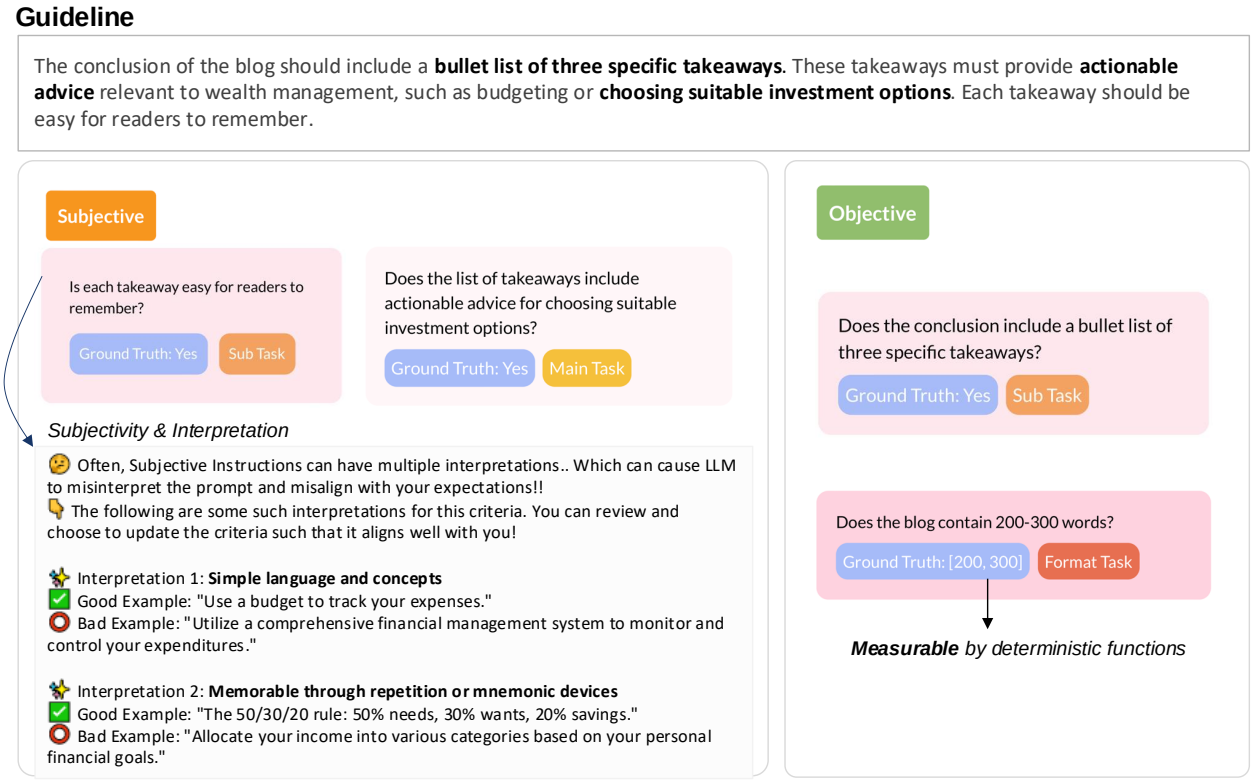

Figure 4: This figure showcases the categorization of instructions as 'Subjective' or 'Objective' by CoPrompter. Objective tasks are clearly defined instructions, whereas subjective instructions can be interpreted in multiple ways. CoPrompter also provides the user with different interpretations of subjective instructions, along with a positive and negative example for each.

(1) Measurable Criteria can be evaluated using deterministic code-based functions without using LLM_{Eval}. Examples include word counts, sentence counts, or checking for a specific keyword. To handle more specific requirements, we add:
   - Layered Measurable Criteria require a two-phase evaluation process. For example, for the guideline: *The conclusion of the response should be under 50 words*, LLM_{Eval} first identifies the conclusion section, then applies a measurable check (word count) to verify its under 50 words.

   The ground truths for measurable criteria are either specific numbers or range of numbers.

(2) Descriptive Criteria cannot be evaluated through simple checks as the other two categories, and are used evaluated directly by LLM_{Eval}. The ground truth for these are binary yes/no answer.

As CoPrompter extracts and breaks down instructions from the user's guidelines, the scope of possible criteria depends on how the user defines them. For instance, a user might specify a 'truthfulness' criterion requiring fact-checking, CoPrompter will establish criteria that judge the output's retrieval capability. Similarly, if user provides a style constraint limiting the use of informal language, the corresponding criteria will judge the style of the output.

In Appendix 4, we compare our criteria-based evaluation approach with a baseline on 10 different prompts from the Dolomites dataset[41] to assess the effectiveness of our approach in covering user guidelines. Our findings show that our method covers 100% of the checks specified in the user guidelines, whereas the baseline achieves only 76.5%, further validating that our criteria-based evaluation ensures comprehensive coverage.

There are additional tags like subjectivity cues and the theme of the criteria (content/style), which further help in seeing a comprehensive alignment report. For subjective criteria, CoPrompter highlights them in the interface (see Figure 4), allowing users to verify and refine the meaning. In the next section, we detail how provide users control to update such criteria.

These details are provided in the Appendix 2. Each criterion is generated using the task objective and atomic instructions from Steps 1 and 2, with prompts detailed in Appendix 8. Metadata for each criterion is generated at the criterion level, with prompt details available in the Appendix 8.

(2) Update Criteria Module

This module allows users to delete, edit, or add criteria and accommodates evolving requirements as seen in Section 5.1.1). Once users *save criteria*, these updates are used in the next evaluation, allowing for iterative refinement as users gain insights through CoPrompter.

(3) Prompt Response Generation Module

This module generates multiple responses based on the user's prompt draft, using LLM_{User} and the number of responses specified by the user.

④ **Alignment Report Card Module**

This module evaluates the generated responses against the saved criteria and presents the results in an interactive report, helping users understand how well the responses align with their requirements.

Determining Alignment. A criterion is considered **aligned** if the LLM_{Eval}'s generated criteria answer matches the user-provided ground truth. We compute the alignment score for each response and criterion pair using the following method:

$$\text{Score} = \begin{cases} 1, & \text{if } \text{LLM}_{\text{Eval}}\text{'s criteria-response answer} \\ & \text{matches ground truth (or within acceptable range)} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We then calculate the *alignment percentage* for each criterion by averaging the scores across all generated responses:

$$\text{Alignment\% for Criterion} = \left(\frac{\sum_{k=1}^{N} \text{Score}_k}{N} \right) \times 100\%$$

where N is the total number of generated responses, and Score_k is the score for response k. This alignment percentage is displayed next to each criterion in the UI, indicating the proportion of responses that met the criterion.

Aggregate Alignment Scores. In the UI, the Alignment Report begins with an overall alignment score, shown as a pie chart with proportion of aligned versus misaligned criteria. This score is calculated as:

$$\text{Overall Alignment Score} = \left(\frac{\text{Criteria with 100\% Alignment}}{\text{Total Criteria}} \right) \times 100\% \tag{2}$$

where criteria with 100% alignment are those where *all* responses received a score of 1.

Next, for each instruction from the guidelines, the report (in UI) shows how many associated criteria are aligned. For example, "1/2" means there are two criteria linked to that instruction, and one of them is aligned. This provides a clear view of alignment at the instruction level.

Finally, at the bottom of the report, we also present alignment scores based on different categories (content or style and subjective or objective) , calculated as:

$$\text{Alignment Score for Category} = \left(\frac{\text{Total Aligned Criteria in Category}}{\text{Total Criteria in Category}} \right) \times 100\% \tag{3}$$

In this manner, we compute alignment and provide users with an interactive report for both prompt responses or their sample outputs.

6 USER EVALUATION

To evaluate how Prompt Engineers make use of CoPrompter, we conducted a user evaluation study. We wanted to find answers to the following research questions inspired by our design goals -

- **(EQ1)** How do Prompt Engineers utilize CoPrompter to improve their prompts?
- **(EQ2)** In what ways does CoPrompter help Prompt Engineers identify misalignments compared to traditional approaches?
- **(EQ3)** What are the key benefits and challenges (both usability and conceptual) of using CoPrompter for prompt improvement?

6.1 Participant Recruitment

We conducted a focus study with 8 industry practitioners experienced in crafting complex prompts (typically over 10 instructions), which allowed us to closely observe how they navigate and use CoPrompter to improve their prompts. Participants were recruited from an IT company using snowball sampling[3], targeting individuals familiar with long-form prompts for models like OpenAI and Llama across tasks such as question answering, content generation, and LLM evaluation. Due to the specialized expertise required for working with complex prompts, which resulted in a limited recruitment pool, and the significant time commitment (75–90 minutes per session), we followed the 5-user rule to efficiently identify most usability issues through overlapping feedback[1]. Study sessions were held via Microsoft Teams, lasting 75–90 minutes to support the iterative nature of prompting. Sessions were recorded to capture prompt histories, interactions, and survey responses, with anonymized data accessible only to the research team. Participation was voluntary, with informed consent obtained in advance.

Among the 8 participants (3 women, 5 men), 4 focused on product engineering use cases, while the other 4 worked in industrial research. All had significant experience with complex prompts to achieve specific outcomes. To ensure anonymity, participants are referred to as P[n] throughout this work.

6.2 Study Design

This study evaluated how prompt engineers use CoPrompter to identify misalignments and improve their prompts while understanding their mental models when interacting with the system compared to traditional prompt refinement methods.

The study had two parts. First, a pre-study interview gathered insights into participants' experiences with complex prompts, their traditional design approaches, and alignment evaluation methods. Participants highlighted that manual prompt creation is time-consuming, often requiring over 10 iterations for response inspection. Due to the impracticality of replicating this process, qualitative comparisons were made through interviews.

In the second part, participants received a demo of CoPrompter and completed a structured task. They defined requirements for a website feature that generates articles using LLMs, identified as a relevant use case in our formative study (Appendix 3). Participants selected their domain, brainstormed article guidelines (5–10 minutes), and used CoPrompter for 40–45 minutes to craft and refine prompts while thinking aloud. Post-task, they provided feedback on CoPrompter 's reliability, workflow integration, challenges, and improvements, and rated their experience using a System Usability Scale (SUS) questionnaire.

[3] https://methods.sagepub.com/foundations/snowball-sampling

6.3 Data Analysis

The recordings from the user evaluation sessions were transcribed. The recording transcriptions were analyzed using Thematic Analysis approach [10]. The transcripts were coded by the research team and the codes were collected, discussed, and categorized over three iterations using the whiteboarding tool Miro [4]. SUS Scores were computed and categorized for a better understanding of user feedback. Our findings are motivated by our data analysis.

7 USER EVALUATION FINDINGS

7.1 How do prompt engineers use CoPrompter to improve LLM Alignment to their Prompts?

As mentioned before, participants were asked to craft their own guidelines for an article generation use case on a topic they liked. The participants chose topics ranging from the need for mental well-being to the importance of investing in mutual funds. An example of a guideline crafted by a user is in Figure 6:

The article needs to be about home security solutions like video doorbells, motion sensors, burglar alarms etc. It should be at least 100 words long. It should have a clear and engaging title reflecting the topic. Start with an attention-grabbing introduction. It should explain why the products are relevant. Use a professional, yet approachable tone that's easy for a general audience to understand. Avoid excessive jargon. There should be obvious takeaways for readers in the article. End with a strong CTA that encourages readers to engage with the company and buy its products. It should not be boring to read.

Figure 5: Prompt Guidelines prepared by P4

The complexity of prompt guidelines ranged from 12 to 20 instructions across participants. They evaluated 14 criteria questions (ranging from 9 to 18) in 75-90 minutes. We observed that our participants mostly followed a consistent user flow of interacting with CoPrompter. In the following section, we answer the EQ1 and EQ3.

7.1.1 ***Building the Evaluator*** *using Criteria Questions:* Once participants finalized their prompt guidelines, they entered them into the 'Build Your Evaluator' Tab, which generated criteria questions based on their input. Participants reviewed these questions to ensure accurate conversion of their guidelines and made edits as needed, such as modifying content, adjusting language, changing ground truth values, or adding/deleting criteria. On average, only 17% of criteria were edited, indicating the tool's usefulness. During the initial review, no edits were made to existing criteria content or language. Participants noted that breaking guidelines into smaller chunks improved their understanding and helped identify missing or updated requirements. For instance, P4 revised an instruction after realizing it required further consideration.

> *One major problem people often face while curating prompts is that they know what they desire in the output, but they cannot break it down into different chunks, different criteria themselves to write a prompt, at least on the first go, so they might have some basic thoughts that at least these guidelines should be followed. But coming up with a bunch of criteria, the way this tool is doing that is, I think, amazing. So even the generation of criteria itself, I think is very helpful for a prompt engineer to look at these aspects and tailor the prompt accordingly.*

At this stage, participants often added new criteria after identifying additional requirements. For example, P6 added a poem at the end of their response, while P3 added checks for hallucinated content. Some deleted redundant or unimportant criteria, like P5, who removed a similar pair: [C1] *Does the article attract photographers and backgrounds?* and [C2] *Does the article attract photographers and backgrounds to engage with the app?*. They also updated ground truth values and refined requirements, such as changing a fixed numerical value to a range. Participants explored other features and reviewed subjective instruction interpretations. However, P2 found the priority tags misaligned with their preferences. After reviewing and editing the criteria, participants expressed confidence in the final list, saved their changes, and proceeded to analyze their prompt. Overall, they appreciated the criteria's alignment with their needs and the customization ease. P5 said—-

> *I really like the feature of modifying a criterion. Because it is particularly helpful in my case like you saw (referring to deleting criteria and adding criteria). So I think that can be like a helpful thing in general also to many people.*

7.1.2 Analysing Prompt Response Alignment: After finalizing the criteria list, participants moved to the 'Analyse Your Prompt' Tab. Some, like P7, added context and structured the prompt, while P3 used bullets and P8 kept the original guidelines, as shown in Figure 6. Participants selected the number of responses to evaluate, ranging from 5 to 10, with a limit of 10 to avoid high latency. They then viewed the Prompt Alignment Report, which included an overall alignment pie chart highlighting misalignments. As participants inspected per-instruction alignment scores, they also opened responses to evaluate if CoPrompter's assessment matched their preferences. The tool's detailed reasoning behind each score was highly appreciated, building trust and transparency in the evaluation process. Some participants expressed concerns about using an LLM as a judge, but the reasoning helped clarify potential evaluation issues. As P3 stated–

> *I think the reasoning column right, I think that would be very useful even in the case of hallucinations. Sometimes the model hallucinates or does not interpret the question as you do, for that reasonings really help.*

As participants noticed misalignments, they edited the prompt in the left panel to adjust instructions or add new requirements based on responses. For instance, P2 observed 0% alignment with a defined word length (50–100 words) and emphasized the instruction

[4] https://miro.com/app

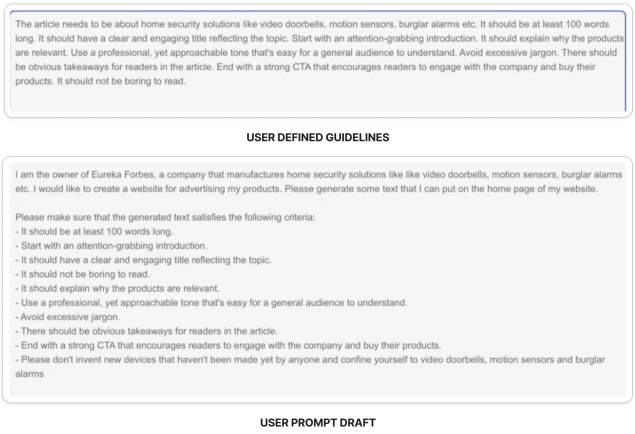

Figure 6: P3's prompt guideline for 'Build Your Evaluator' Tab and prompt draft for 'Analyse Your Prompt' Tab

by adding phrases like "Make sure."

Participants noticed discrepancies between alignment scores and their evaluations of responses. By reviewing score justifications, they identified changes needed for criteria questions and updated the 'Build Your Evaluator' tab. For example, P5 revised *'Does the response engage photographers from all ages and backgrounds?'* to *'Does the article mention how the app would be relevant for older photographers while also highlighting features for younger photographers?'* to better reflect their requirements. P7 also added *'Does the article contain emoticons?'* after noticing their absence in responses.

Participants re-evaluated their updated prompts, some increasing the number of responses. P1 noted that more responses might increase misalignment, prompting further testing. Overall, alignment scores improved, and participants confirmed responses better met their needs.

Some used the 'Analyse Sample Outputs' Tab to explore CoPrompter 's capabilities. P8 fed a ChatGPT-generated prompt as a sample response, while P3 used the tab to test CoPrompter with undesirable responses after a high alignment in their crafted prompt.

7.2 CoPrompter as a *Helping Hand* in Prompt Improvement Workflows

The following section answers the EQ2. Overall, participants expressed that CoPrompter can be a useful addition to their prompt improvement workflows. P3 established -

> *Basically, I manually check the effects of the changes that I made in the prompt, but if I have a helping hand from a tool like CoPrompter, I can check the scores, edit the prompt, and recheck the scores that it has provided, and improve where the score is low.*

We observed the following broad themes on the participant user experience with CoPrompter:

7.2.1 Criteria Generation - Enabling Control over Evaluation and Evolving Requirements. Participants found the criteria list a great ***starting point*** as a heuristic, validating findings from literature [59]. They appreciated the flexibility to modify criteria during the evaluation process. Reviewing the list or alignment scores often revealed additional requirements they hadn't initially considered, providing greater control and allowing the criteria to evolve alongside their insights and needs. This adaptability also facilitated the discovery of new requirements. As a participant reflected–

> *So even if I keep aside the second part where the alignment scores are shown, the generation of criteria itself, I think is very helpful for prompt engineers as they can look at these granular aspects of their requirements and tailor the prompts accordingly. And of course it comes with the added advantage and the icing on the cake, which is a LLM also scoring your responses on these criteria. So it is like the* ***next level of automation****.*

Overall, all the participants approved using the generated criteria list as an evaluation metric after making edits to it.

7.2.2 Alignment Report Card – Streamlining Prompt Refinement Compared to Conventional Methods. Participants compared CoPrompter to traditional prompt improvement methods. They recalled that models often fail at tasks with format constraints, overlook specific instructions, or misinterpret conditions. Despite available prompt engineering guides, engineers typically use trial-and-error to check alignment. Moreover, modifying one instruction could disrupt the alignment of others, requiring constant rechecking. As a participant said-

> *Many times, I don't know why I'm doing what I am doing when I am making changes in my prompts, but I just try to see if the responses get better, For example, I start trying some random things or just changing the order of instructions. like there's a lot of trial and error.*

Participants used CoPrompter to pinpoint changes needed for better alignment. As P1 highlighted, CoPrompter clarified *what's working* and *what's not working* in the prompt. While some disagreed with evaluation scores, the reasoning helped them understand and adjust the evaluation and prompt, making the process more *systematic* and *clean*. Overall, participants felt CoPrompter would be a valuable addition to their prompt improvement workflows. As P2 said–

> *Traditionally, It's more about trial and error, but it's not structured. So I think adding the structure to it right and having a structured way of analyzing the prompt that - actually helps out much more than just trial and error or even asking the model to optimize a prompt. So just the structure itself makes it easier to find out which part of the prompt we need to modify.*

Manual inspection of responses to identify misalignments was a common challenge. While some used quantitative checks for response format in their traditional pipeline, manual verification of other instructions was still required. CoPrompter improved this by offering a one-stop solution for checking misalignments. As P3 recalled–

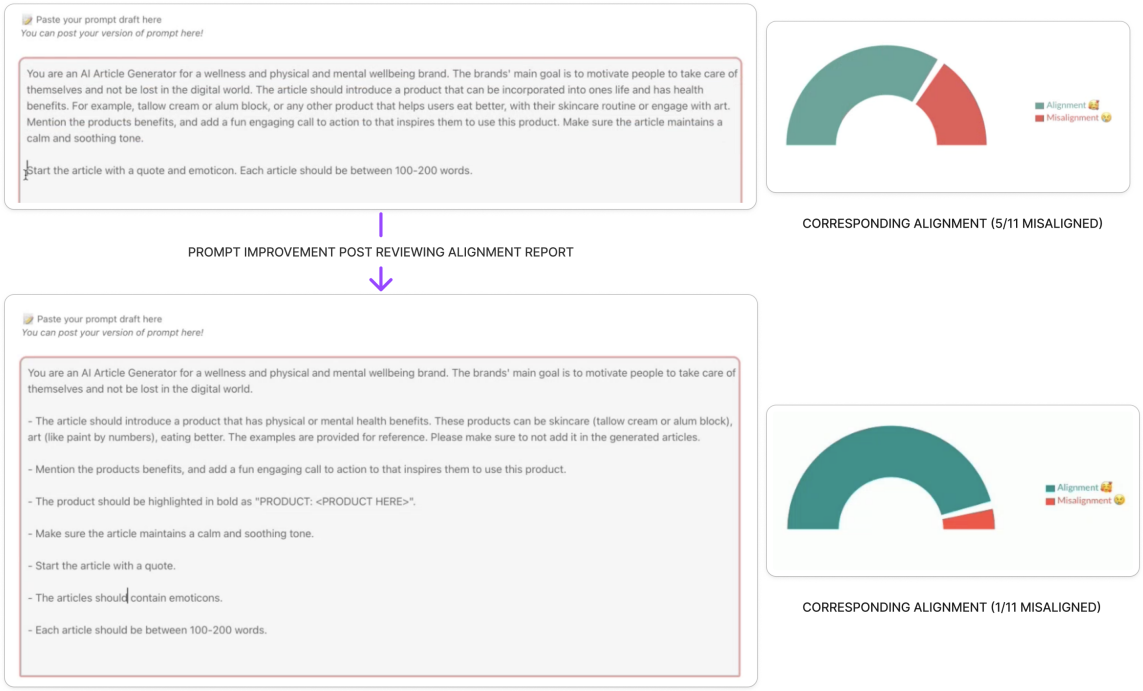

Figure 7: P7 alignment performance improved on making prompt edits (adding instructions, changing format, updating older instructions based on alignment scores) and updating the criteria list to align it better with their requirements.

> *I just edit my prompt somewhere in the middle of a Python file and then start a server and test it for many responses. And then dump them in some file, and then read those responses manually. And so basically I don't have one UI or tool for this, I have to do everything in the IDE basically which becomes chaotic.*

Participants appreciated that CoPrompter automated alignment checks across responses, eliminating the need for time-consuming manual verification. By breaking prompts into criteria and evaluating them systematically, CoPrompter streamlined the process and saved significant time. A participant reflected -

> *It's like usually hard to manually inspect like, say 15 criteria on say 20 or 30 responses right? So that's a big problem that's solved, it will save a lot of time.*

Participants appreciated the tool's intuitive and easy-to-follow interface. The user flow mirrored their usual prompt improvement process, including using guidelines to create prompts and manually evaluating multiple responses. They also valued the tags and color coding, which helped them better understand their instructions.

> *A lot of the things in this tool are very easy to follow. I mean, initially when a new user comes to this interface, obviously there are going to be some questions, but if you consider a user who is going to be regularly using this. I think things are really, really easy to follow and adapt for one's own use cases. So I think that is really commendable. I notice the use of specific colours to indicate different things and all that actually stands out in the subconscious mind, and this is something that definitely is worth appreciating.*

Participants identified nontrivial applications of CoPrompter based on their use cases. P3 suggested using it to reduce human annotation effort by focusing on the criteria review stage. P1 highlighted its potential to enhance model interpretability by pinpointing instructions the model struggles with. Participants also expressed interest in extending CoPrompter to text-image models for image evaluation. P7 proposed using it to document prompt evolution and set alignment scores as standards for prompt quality. Overall, they found quantifying alignment relevant and intriguing.

7.3 Painpoints and Suggestions

7.3.1 Challenges in navigating CoPrompter. This section answers EQ3. Participants found CoPrompter user-friendly but were initially hesitant about using LLMs for certain evaluations, with some expressing distrust. However, features like viewing responses, understanding alignment scores, and analyzing sample responses helped build trust in the evaluation process.

P1 and P2 noted CoPrompter may struggle with objective tasks like code evaluation, recommending a chunk-based approach with criteria lists. Some struggled to understand tag reasoning for criteria questions; P4 found "it should sound exciting" ambiguous, and P8 noted unclear distinctions between Objective and Subjective tags. P1 suggested improving explainability and guides, though others disagreed.

P8 observed that the significance of each instruction varies and suggested incorporating relative weight to generate a more accurate alignment score based on the prompt engineer's priorities.

Participants experienced wait times up to a minute due to API call latency, which P1 referred to as the "cost" of CoPrompter's ease. Despite this, participants remained patient.

All except P8 said they could integrate CoPrompter into their prompt improvement pipelines. P8 preferred receiving alignment reports via CLI and expressed interest in an SDK version to streamline onboarding.

7.3.2 Suggestions and Points of Improvement. Participants suggested adding prompt improvement suggestions using LLMs and CoPrompter scores. They believed this would make the process easier by providing actionable suggestions corresponding to alignment scores. P4 proposed guideline suggestions based on task intent, helping users avoid missing obvious or implicit guidelines. P1 suggested offering prompt templates to create effective prompts. Overall, participants wanted actionable ways to improve prompts after identifying areas for improvement.

P3 suggested using reasoning in prompt alignment reports to detect hallucinations if grounded in user-provided documents. Lastly, P8 recommended highlighting misalignments during prompt creation, similar to how grammar errors are flagged in Grammarly[5], to save time during the review process.

7.4 System Usability Scale (SUS) Survey

Participants completed a SUS Survey with 10 questions, split into System Usability (user-friendliness) and System Efficiency (goal achievement), with 5 questions in each category. The full list of questions is in the appendix.

Figure 8 shows that 45% of users were "satisfied" and 40% "very satisfied" with System Usability, reflecting positive feedback and frequent use of CoPrompter. For System Efficiency, 50% rated it "very satisfied" and 40% "satisfied," indicating ease of use and quick learning.

The mean SUS score was 81.25, placing CoPrompter in the "A" range of excellence according to Sauro and Lewis's CGS [55] (Figure 9). This score is above the industry average of 68 ("Okay") [54], indicating above-average usability. Despite P1's dissatisfaction with system complexity (blue bar in Figure 8), the high overall score reflects the tool's efficiency, user-friendliness, and quick task completion.

[5]https://app.grammarly.com/

8 DISCUSSION

8.1 CoPrompter as an assistant for Prompt Improvement

Our user evaluation showed that CoPrompter helps prompt engineers systematically address instruction-following misalignments **(DG1)**. Participants preferred it to manual methods, praising the ability to evaluate multiple responses efficiently, check user requirements comprehensively, and generate detailed alignment reports. The user-in-loop approach ensures evaluations reflect the user's alignment definitions.

Participants also highlighted the ability to refine prompts by pinpointing misaligned instructions that improved the traditional trial-and-error methods **(DG2)**. Additionally, CoPrompter supports evolving evaluation needs **(DG3)** by allowing dynamic updates to the Evaluation Criteria List, enabling flexibility as new requirements arise during the review process.

While CoPrompter achieves the goal of streamlining misalignment identification and prompt improvement, participants noted that it could help clarify their requirements by breaking down guidelines into granular instructions, aiding in drafting prompts that accurately capture intent. Furthermore, they wanted to see a categorical analysis of instructions the model struggles with. Participants also mentioned CoPrompter potential for tracking prompt and alignment score evolution for monitoring and presentation and act as a human annotator by conducting human-aligned evaluations of responses.

8.2 Considerations for designing for Human LLM Alignment Workflows

8.2.1 Alignment is Dynamic. The concept of Human-LLM alignment has evolved from a uni-directional model to a *bi-directional framework*, where Human alignment to AI and AI alignment to Human are interconnected [61]. Terry et al. break down alignment into specification, process, and evaluation [64]. Alignment is dynamic, varying across users and evolving over time. As seen in our study, users have unique definitions of 'response alignment,' highlighting the need for a *user-centric approach* in Human-AI Alignment systems. CoPrompter supports this by giving users control over the evaluation process, allowing them to define alignment for their specific use case.

Furthermore, user alignment expectations can change as their prompts evolve. Our study showed users encountered new requirements, and Human-AI Alignment systems must account for this *evolution of alignment*. CoPrompter enables users to modify the evaluation pipeline at any time. We recommend that future systems consider the dynamic nature of alignment and leverage user-driven approaches, incorporating human aspirations, values, and assets. These factors can be explored using human-centric research methods like participatory design, value-sensitive design, and asset-based design [14, 52, 62].

8.2.2 Trust as a Key Driver in Alignment Evaluation Workflows. Human trust in LLMs is crucial in Human-AI Interaction [3, 9, 19, 65]. We observed users' mistrust when learning about LLMs' role in

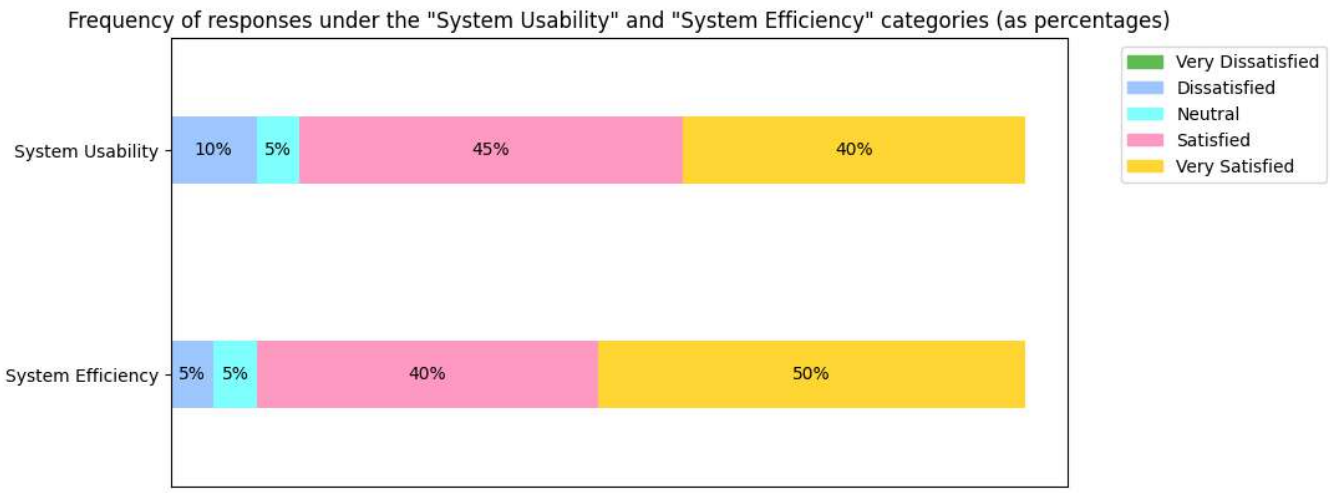

Figure 8: Participant ratings on System Usability and Efficiency

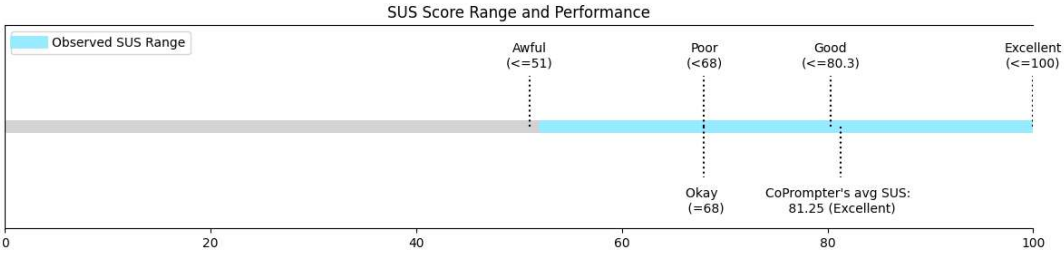

Figure 9: SUS Score Range and Aggregate across participants

alignment evaluation, as LLMs can be biased, lack nuance, and misinterpret context [47]. However, CoPrompter addressed this mistrust by providing reasoning behind alignment scores, allowing users to understand and evaluate the rationale. Participants also tested CoPrompter with custom responses to compare alignment scores with their own judgments, which helped build trust and transparency. This approach improved users' interpretation of LLM behavior and highlighted the importance of user trust in building reliable Alignment Evaluation Workflows. Since alignment is user-defined, it's critical for users to understand the rationale behind alignment judgments.

8.3 Limitations and Future Work

CoPrompter helps prompt engineers refine their prompts by identifying misalignments between LLM outputs and user-defined criteria, with positive feedback from industry professionals and high System Usability Scale (SUS) scores demonstrating its effectiveness. This work shows that with the right tools and methods, we can improve Human-AI alignment and make LLM applications more reliable across different domains. However, this work also poses certain limitations, which are opportunities for future work.

(1) The evaluation was conducted in a focus study format with a small sample of eight participants, which may limit generalizability. Expanding to larger, real-world studies will provide deeper insights into diverse user behaviors and applications.

(2) The current binary scoring mechanism, while effective, may also oversimplify complex user guidelines, limiting the granularity of evaluation. Adopting a scale-based system could address this limitation by capturing finer gradients of alignment per criterion.

(3) Additionally, LLM-driven evaluation systems exhibit several biases such as sociocultural, political, and context insensitivity [15, 28, 35], which may skew assessments by misinterpreting context or reflecting underlying training data biases.

Lastly, CoPrompter could be improved to have functionalities such as fact-checking, hallucination detection, or suggesting prompt improvements from alignment reports.

REFERENCES

[1] Roobaea Alroobaea and Pam J. Mayhew. 2014. How many participants are really enough for usability studies?. In *2014 Science and Information Conference*. 48–56. https://doi.org/10.1109/SAI.2014.6918171

[2] Ian Arawjo, Priyan Vaithilingam, Martin Wattenberg, and Elena Glassman. 2023. ChainForge: An open-source visual programming environment for prompt engineering. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–3.

[3] Patrick Bedué and Albrecht Fritzsche. 2021. Can We Trust AI? An Empirical Investigation of Trust Requirements and Guide to Successful AI Adoption. *Journal of Enterprise Information Management* 35 (04 2021). https://doi.org/10.1108/JEIM-06-2020-0233

[4] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 610–623.

[5] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv. *arXiv preprint arXiv:2303.12712* (2023).

[8] Yi Chen, Rui Wang, Haiyun Jiang, Shuming Shi, and Ruifeng Xu. 2023. Exploring the Use of Large Language Models for Reference-Free Text Quality Evaluation: An Empirical Study. arXiv:2304.00723 [cs.CL] https://arxiv.org/abs/2304.00723

[9] Oscar Hengxuan Chi, Shizhen Jia, Yafang Li, and Dogan Gursoy. 2021. Developing a formative scale to measure consumers' trust toward interaction with artificially intelligent (AI) social robots in service delivery. *Computers in Human Behavior* 118 (2021), 106700. https://doi.org/10.1016/j.chb.2021.106700

[10] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *The Journal of Positive Psychology* 12, 3 (2017), 297–298. https://doi.org/10.1080/17439760.2016.1262613 arXiv:https://doi.org/10.1080/17439760.2016.1262613

[11] Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and Unfairness in Information Retrieval Systems: New Challenges in the LLM Era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6437–6447.

[12] Carson Denison, Monte MacDiarmid, Fazl Barez, David Duvenaud, Shauna Kravec, Samuel Marks, Nicholas Schiefer, Ryan Soklaski, Alex Tamkin, Jared Kaplan, et al. 2024. Sycophancy to Subterfuge: Investigating Reward-Tampering in Large Language Models. *arXiv preprint arXiv:2406.10162* (2024).

[13] Leonard Dung. 2023. Current cases of AI misalignment and their implications for future risks. *Synthese* 202, 5 (26 Oct 2023), 138. https://doi.org/10.1007/s11229-023-04367-0

[14] Batya Friedman, David G. Hendry, and Alan Borning. 2017. A Survey of Value Sensitive Design Methods. *Found. Trends Hum.-Comput. Interact.* 11, 2 (Nov. 2017), 63–125. https://doi.org/10.1561/1100000015

[15] Isabel O. Gallegos, Ryan A. Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K. Ahmed. 2024. Bias and Fairness in Large Language Models: A Survey. *Computational Linguistics* 50, 3 (Sept. 2024), 1097–1179. https://doi.org/10.1162/coli_a_00524 Place: Cambridge, MA Publisher: MIT Press.

[16] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 3356–3369. https://doi.org/10.18653/v1/2020.findings-emnlp.301

[17] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462* (2020).

[18] David Glukhov, Ilia Shumailov, Yarin Gal, Nicolas Papernot, and Vardan Papyan. [n. d.]. Position: Fundamental Limitations of LLM Censorship Necessitate New Approaches. In *Forty-first International Conference on Machine Learning*.

[19] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 8342–8360. https://doi.org/10.18653/v1/2020.acl-main.740

[20] Jeffrey Heer and Ben Shneiderman. 2012. Interactive Dynamics for Visual Analysis: A taxonomy of tools that support the fluent and flexible use of visualizations. *Queue* 10, 2 (Feb. 2012), 30–55. https://doi.org/10.1145/2133416.2146416

[21] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689* (2022).

[22] Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2022. Can Large Language Models Truly Follow your Instructions?. In *NeurIPS ML Safety Workshop*.

[23] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. Towards Mitigating LLM Hallucination via Self Reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 1827–1843. https://doi.org/10.18653/v1/2023.findings-emnlp.123

[24] Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. Promptmaker: Prompt-based prototyping with large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–8.

[25] Fengqing Jiang, Zhangchen Xu, Luyao Niu, Boxin Wang, Jinyuan Jia, Bo Li, and Radha Poovendran. 2024. POSTER: Identifying and Mitigating Vulnerabilities in LLM-Integrated Applications. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*. 1949–1951.

[26] Ling Jiang, Keer Jiang, Xiaoyu Chu, Saaransh Gulati, and Pulkit Garg. 2024. Hallucination Detection in LLM-enriched Product Listings. In *Proceedings of the Seventh Workshop on e-Commerce and NLP @ LREC-COLING 2024*, Shervin Malmasi, Besnik Fetahu, Nicola Ueffing, Oleg Rokhlenko, Eugene Agichtein, and Ido Guy (Eds.). ELRA and ICCL, Torino, Italia, 29–39. https://aclanthology.org/2024.ecnlp-1.4

[27] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410* (2023).

[28] Ishika Joshi, Ishita Gupta, Adrita Dey, and Tapan Parikh. 2024. 'Since Lawyers are Males..': Examining Implicit Gender Bias in Hindi Language Generation by LLMs. https://doi.org/10.48550/arXiv.2409.13484 arXiv:2409.13484 [cs].

[29] 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 CHI Conference on Human Factors in Computing Systems*. 1–21.

[30] Yoonsu Kim, Kihoon Son, Seoyoung Kim, and Juho Kim. 2024. Beyond Prompts: Learning from Human Communication for Enhanced AI Intent Alignment. *arXiv preprint arXiv:2405.05678* (2024).

[31] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Aaron Jiaxun Li, Soheil Feizi, and Himabindu Lakkaraju. 2024. Certifying LLM Safety against Adversarial Prompting. https://openreview.net/forum?id=wNere1lelo

[32] Yingjie Li, Qidong Yan, Ning Ma, and Fucheng Wan. 2024. Research on Alignment and Evaluation Methods for Large Language Models. In *Proceedings of the 2024 Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Digital Economy and Artificial Intelligence* (Hongkong, China) *(DEAI '24)*. Association for Computing Machinery, New York, NY, USA, 710–715. https://doi.org/10.1145/3675417.3675536

[33] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[34] Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 3214–3252. https://doi.org/10.18653/v1/2022.acl-long.229

[35] Andy Liu, Mona Diab, and Daniel Fried. 2024. Evaluating Large Language Model Biases in Persona-Steered Generation. In *Findings of the Association for Computational Linguistics: ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 9832–9850. https://doi.org/10.18653/v1/2024.findings-acl.586

[36] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. Calibrating LLM-Based Evaluator. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (Eds.). ELRA and ICCL, Torino, Italia, 2638–2656. https://aclanthology.org/2024.lrec-main.237

[37] Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2024. HD-Eval: Aligning Large Language Model Evaluators Through Hierarchical Criteria Decomposition. *arXiv preprint arXiv:2402.15754* (2024).

[38] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*. PMLR, 22631–22648.

[39] Manikanta Loya, Divya Sinha, and Richard Futrell. 2023. Exploring the Sensitivity of LLMs' Decision-Making Capabilities: Insights from Prompt Variations and Hyperparameters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 3711–3716. https://doi.org/10.18653/v1/2023.findings-emnlp.241

[40] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 8086–8098. https://doi.org/10.18653/v1/2022.acl-long.556

[41] Chaitanya Malaviya, Priyanka Agrawal, Kuzman Ganchev, Pranesh Srinivasan, Fantine Huot, Jonathan Berant, Mark Yatskar, Dipanjan Das, Mirella Lapata, and Chris Alberti. 2024. DOLOMITES: Domain-Specific Long-Form Methodical Tasks. *arXiv preprint arXiv:2405.05938* (2024).

[42] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*. https://openreview.net/forum?id=RwzFNbJ3Ez

[43] Zeeshan Memon, Muhammad Arham, Adnan Ul-Hasan, and Faisal Shafait. 2024. LLM-Informed Discrete Prompt Optimization. In *ICML 2024 Workshop on LLMs and Cognition*. https://openreview.net/forum?id=d0jQuZe6k0

[44] Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, Online, 5356–5371. https://doi.org/10.18653/v1/2021.acl-long.416

[45] Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. 2024. Mitigating Hallucinations in Large Language Models via Self-Refinement-Enhanced Knowledge Retrieval. In *The Second Workshop on Generative Information Retrieval*. https://openreview.net/forum?id=H6Kz3tRugR

[46] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=TG8KACxEON

[47] Qian Pan, Zahra Ashktorab, Michael Desmond, Martín Santillán Cooper, James Johnson, Rahul Nair, Elizabeth Daly, and Werner Geyer. 2024. Human-Centered Design Recommendations for LLM-as-a-judge. In *Proceedings of the 1st Human-Centered Large Language Modeling Workshop*, Nikita Soni, Lucie Flek, Ashish Sharma, Diyi Yang, Sara Hooker, and H. Andrew Schwartz (Eds.). ACL, TBD, 16–29. https://doi.org/10.18653/v1/2024.hucllm-1.2

[48] Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint Instruction Following for Long-form Text Generation. arXiv:2406.19371 [cs.CL] https://arxiv.org/abs/2406.19371

[49] Mansi Phute, Alec Helbling, Matthew Daniel Hull, ShengYun Peng, Sebastian Szyller, Cory Cornelius, and Duen Horng Chau. 2024. LLM Self Defense: By Self Examination, LLMs Know They Are Being Tricked. In *The Second Tiny Papers Track at ICLR 2024*. https://openreview.net/forum?id=YoqgcIA19o

[50] Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. InFoBench: Evaluating Instruction Following Ability in Large Language Models. arXiv:2401.03601 [cs.CL] https://arxiv.org/abs/2401.03601

[51] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[52] Smirla Ramos Montañez. 2023. Advancing equity through research: The importance of asset-based approaches and methods. *Journal of Applied Developmental Psychology* 86 (2023), 101540. https://doi.org/10.1016/j.appdev.2023.101540

[53] Abhinav Sukumar Rao, Atharva Roshan Naik, Sachin Vashistha, Somak Aditya, and Monojit Choudhury. 2024. Tricking LLMs into Disobedience: Formalizing, Analyzing, and Detecting Jailbreaks. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (Eds.). ELRA and ICCL, Torino, Italia, 16802–16830. https://aclanthology.org/2024.lrec-main.1462

[54] J. Sauro. 2011. *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Measuring Usability LLC. https://books.google.co.in/books?id=BL0kKQEACAAJ

[55] J. Sauro and James Lewis. 2012. *Quantifying the User Experience*. https://doi.org/10.1016/C2010-0-65192-3

[56] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=RIu5lyNXjT

[57] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, JD Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G Parameswaran, and Eugene Wu. 2024. Spade: Synthesizing assertions for large language model pipelines. *arXiv preprint arXiv:2401.03038* (2024).

[58] Shreya Shankar, Haotian Li, Parth Asawa, Madelon Hulsebos, Yiming Lin, J. D. Zamfirescu-Pereira, Harrison Chase, Will Fu-Hinthorn, Aditya G. Parameswaran, and Eugene Wu. 2024. SPADE: Synthesizing Data Quality Assertions for Large Language Model Pipelines. arXiv:2401.03038 [cs.DB] https://arxiv.org/abs/2401.03038

[59] Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya G Parameswaran, and Ian Arawjo. 2024. Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. *arXiv preprint arXiv:2404.12272* (2024).

[60] Mrinank Sharma, Meg Tong, Tomasz Korbak, David Duvenaud, Amanda Askell, Samuel R Bowman, Newton Cheng, Esin Durmus, Zac Hatfield-Dodds, Scott R Johnston, et al. 2023. Towards understanding sycophancy in language models. *arXiv preprint arXiv:2310.13548* (2023).

[61] Hua Shen, Tiffany Knearem, Reshmi Ghosh, Kenan Alkiek, Kundan Krishna, Yachuan Liu, Ziqiao Ma, Savvas Petridis, Yi-Hao Peng, Li Qiwei, et al. 2024. Towards Bidirectional Human-AI Alignment: A Systematic Review for Clarifications, Framework, and Future Directions. *arXiv preprint arXiv:2406.09264* (2024).

[62] Clay Spinuzzi. 2005. The Methodology of Participatory Design. *Technical Communication* 52 (05 2005), 163–174.

[63] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

[64] Michael Terry, Chinmay Kulkarni, Martin Wattenberg, Lucas Dixon, and Meredith Ringel Morris. 2023. Interactive AI Alignment: Specification, Process, and Evaluation Alignment. https://api.semanticscholar.org/CorpusID:264935292

[65] Takane Ueno, Yuto Sawa, Yeongdae Kim, Jacqueline Urakami, Hiroki Oura, and Katie Seaborn. 2022. Trust in Human-AI Interaction: Scoping Out Models, Measures, and Methods. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 254, 7 pages. https://doi.org/10.1145/3491101.3519772

[66] Yixin Wan, George Pu, Jiao Sun, Aparna Garimella, Kai-Wei Chang, and Nanyun Peng. 2023. "Kelly is a Warm Person, Joseph is a Role Model": Gender Biases in LLM-Generated Reference Letters. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 3730–3748. https://doi.org/10.18653/v1/2023.findings-emnlp.243

[67] Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. 2024. Do-Not-Answer: Evaluating Safeguards in LLMs. In *Findings of the Association for Computational Linguistics: EACL 2024*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, St. Julian's, Malta, 896–911. https://aclanthology.org/2024.findings-eacl.61

[68] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705* (2022).

[69] Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. 2024. SELF-GUARD: Empower the LLM to Safeguard Itself. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 1648–1668. https://doi.org/10.18653/v1/2024.naacl-long.92

[70] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[71] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359* (2021).

[72] Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and Prejudice: LLM Amplifies Self-Bias in Self-Refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 15474–15492.

[73] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. Re-examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376301

[74] Kai-Ching Yeh, Jou-An Chi, Da-Chen Lian, and Shu-Kai Hsieh. 2023. Evaluating interfaced llm bias. In *Proceedings of the 35th Conference on Computational Linguistics and Speech Processing (ROCLING 2023)*. 292–299.

[75] 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* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

[76] Mingyuan Zhang, Zhaolin Cheng, Sheung Ting Ramona Shiu, Jiacheng Liang, Cong Fang, Zhengtao Ma, Le Fang, and Stephen Jia Wang. 2023. Towards Human-Centred AI-Co-Creation: A Three-Level Framework for Effective Collaboration between Human and AI. In *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing* (Minneapolis, MN, USA) *(CSCW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 312–316. https://doi.org/10.1145/3584931.3607008

[77] Zhiping Zhang, Michelle Jia, Hao-Ping Lee, Bingsheng Yao, Sauvik Das, Ada Lerner, Dakuo Wang, and Tianshi Li. 2024. "It's a Fair Game", or Is It? Examining How Users Navigate Disclosure Risks and Benefits When Using LLM-Based Conversational Agents. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–26.

APPENDIX

1 TYPES OF ADDED CRITERIA

Users can add criteria which are of the following types:

- Measurable: If users want shorter responses, they might add a criterion specifying a word count, either as an exact number (e.g., =200 words) or an acceptable range (e.g., [200, 300] words).
- Descriptive: Users may want to check for specific type of keywords or if the conclusion contains bullet points, with a simple yes/no ground truth.
- Layered Measurable allows users to set complex evaluation criteria that first identify a specific section (e.g., the conclusion) and then apply a measurable check, such as, *Does the conclusion have fewer than 50 words?* This approach enables targeted evaluation of specific parts of a prompt response.

2 ADDITIONAL CRITERIA METADATA

(1) **Subjectivity Tag**: This attribute represents whether the atomic instruction is Subjective or Objective. Subjective terms are defined as words or phrases that can be interpreted in multiple ways. We added this attribute to highlight to the users that a particular atomic instruction might need special attention due to the present subjectivity. If a particular instruction is tagged as subjective, we also add different interpretations of the instruction along with a positive example and a negative example of the particular interpretation. Based on the interpretations and examples, the users can acquire a clear understanding of how to specify their requirements in the prompt in a more objective manner.

In Figure 4, the instruction, 'The conclusion of the blog should include a bullet list of three specific takeaways.' is labeled as objective, as it provides a clear requirement without any ambiguity. The instruction specifies a format (a bullet list), a fixed number (three takeaways), and a section (the conclusion). Meanwhile, the instruction that each takeaway should be "easy for the user to remember " is tagged as subjective. The phrase "easy to remember" can vary widely in interpretation, leading to different approaches in the generated responses. The first interpretation suggests simplicity in language and concepts, offering clear, straightforward advice like using a budget to track expenses. The second interpretation focuses on memorability through repetition or mnemonic devices, such as the 50/30/20 rule for budgeting. By providing examples for both interpretations, the user can see how the instruction might be understood in different ways, potentially causing misalignment in response generation. This helps the user refine the instruction to better match their intent.

To avoid very close interpretations, we also generate a similarity score for interpretations on a scale from 1 to 5. 1 indicates that the interpretations are very different, and 5 indicates that they are very similar. This way, the generated interpretations can be filtered if needed.

(2) **Question Theme** This attribute explains the theme of the criteria question. The two different tags for the attribute are *Content*, *Style*. Criteria questions that check the qualitative

style or overall format of the response will be of the *Style* theme, for example, tone, vibe, or output format. Criteria questions that check the content of the response, that is, if certain content information is present or if the content task is followed well, will tagged as *Content*.

3 FORMATIVE STUDY QUESTIONNAIRE

The formative study collected insights from participants regarding their experiences, challenges while prompt engineering. Responses reflect the number of users who chose each option, providing a quantitative view of common practices and issues encountered in various application contexts.

(Q) In what context have you previously developed tools/applications using finetuned LLMs? (Select all that apply)

- Customer Facing Products: 17
- Company Hackathons: 14
- Personal Applications: 10
- Internal Products: 6
- Non-customer facing products: 1
- Company Workshops: 1
- POC development: 1

(Q) What kind of tasks have you used LLMs for in the above-mentioned tool? (Select all that apply)

- Knowledge Extraction/ Information Retrieval: 20
- Content Generation: 19
- Conversational Support through chatbots: 15
- Summarization: 15
- Personalisation: 3
- API Execution to perform tasks in the application: 1
- Instruct-guided editing workflows: 1
- Assistant in using the product - executing tasks for the user: 1
- Make charts and graphs: 1
- Question Answering: 1
- Text Classification: 1

(Q) How do you often structure your instructions in your prompt to the LLM for your applications? (Select all that apply)

- Instructions supported with Examples: 21
- List of to-the-point and concise Instructions: 20
- List of detailed and verbose Instructions: 16
- Instructions using content from multiple LLM calls: 11
- Step-by-step demonstrations of how to generate response: 10
- Unstructured Instructions (paragraph format): 10
- Pseudocode instructions using SudoLang: 1
- Enriched RAG with KG query. Semantic search filtering: 1
- Optimizing prompts prior to use in a production tool: 1

(Q) What are the different types of instructions you usually give in a prompt? (Select all that apply)

- Instructions about the objective task to be implemented: 26
- Instructions about the desired format of response: 25
- Instructions about the context of the conversation: 25
- Instructions about the desired style of response: 14
- Interface-oriented, constraint-based programming to express complex behaviors: 1

(Q) What type of misalignments do you observe in the response generated by the LLM? (Select all that apply)

- Some instruction overlooked/ignored: 25
- Hallucinations: 21
- Misinterpretation of instructions: 18
- Assumptions/elements not instructed: 14
- Inconsistent instruction following: 14
- Overfitting to examples: 12
- Incomplete response: 7
- Misinterpretation of Time Zones: 1
- Irrelevant response: 1
- Categorization issues and failure to follow rules: 1
- Failure in tool usage and format consistency: 1

(Q) Which types of instructions do you observe misalignment issues in? (Select all that apply)

- Complex, elaborate instructions: 19
- Format Instructions: 15
- Objective Instructions for conditional actions: 15
- Objective Instructions for content generation: 14
- Subjective Instructions: 7
- Style Instructions: 6
- Instructions with Examples: 5
- Manipulation of discrete data: 1
- API-related chat questions: 1
- Centered text when using examples: 1
- Missing or unclear instructions: 1

(Q) When you encounter misalignment, what strategies do you use to get a better response? (Select all that apply)

- Identifying misaligned instructions: 21
- Providing more context: 19
- Testing over multiple responses: 19
- Hit-and-trial improvements: 18
- Adding new instructions: 17
- Breaking tasks into steps: 14
- Reordering instructions: 14
- Adding more examples: 10
- Voyager technique: 1
- Adding few examples cautiously: 1
- Small datasets or vector database issues: 1
- Reinforcing attention via Local COT: 1
- Identifying ambiguities: 1

(Q) From the above context, which of the following modifications often improve response alignment? (Select all that apply)

- Providing more context: 16
- Adding new instructions: 15
- Breaking tasks into steps: 14
- Hit-and-trial language improvements: 10
- Reordering instructions: 10
- Testing multiple responses: 10
- Simplifying complex prompts: 1
- Adding rare examples: 1
- Upgrading to a newer model: 1
- Inconsistently effective: 1
- Restarting project: 1
- Example input-output tests: 1
- Identifying ambiguities: 1

(Q) What challenges do you face when trying to improve response alignment? (Select all that apply)

- Prompt refinement takes time: 20
- Inconsistent instruction-following: 17
- Unexpected response behavior: 17
- Lack of modification clarity: 7
- Choosing suitable LLM model: 4
- Difficulty expressing intent: 4
- Limited LLM capacity or attention heads: 1
- Misalignment trade-offs in output: 1
- Category issues: 1
- Token and retrieval limitations: 1

(Q) Which LLM do you use often for these use cases?

- OpenAI Models: 27
- Other Large Language Models: 6
- LLama Models: 5
- Custom fine-tuned Models: 4
- GPT-4, Claude 3 Opus, Llama 3 70b, Mixtral 8x7b: 1
- Exploring Claude 3, Mistral: 1
- Mistral/Mixtral: 1
- Vision: 1
- Sonnet (personal projects): 1

(Q) How many instructions do you usually use in your prompts?

- More than 10 instructions: 10
- 5-10 instructions: 8
- 2-5 instructions: 5
- 1 instruction: 4
- Varies (3 to more than 10): 1

(Q) How many attempts does it usually take to design a prompt for the desired response?

- More than 10 attempts: 14
- 4-6 attempts: 6
- 2-3 attempts: 4
- 7-10 attempts: 3
- 1 attempt: 1

(Q) On a scale of 1-5, how aligned is the LLM response to your instructions on the first attempt?

- Rating 3 - 12
- Rating 4 - 11
- Rating 5 - 12

(Q) How many generated responses do you test your prompt on?

- More than 40 responses: 14
- 20-40 responses: 5
- Less than 20 responses: 4

(Q) What are the inputs you use while testing/evaluating your prompt?

- Manually created dataset: 8
- Sample inputs + dataset: 5
- Sample inputs only: 3
- RITEWay test suite: 1
- Various dataset combinations: 1

(Q) Which stages do you check for alignment to prompts?

- Prototype: 11
- Prototype + Production: 6
- All Stages (Prototype, Production, Post-deployment): 1

(Q) Please tell us more about your instruction styles and instruction categories used in your prompts here. Various responses, including single-shot, few-shot, multi-agent architectures, constraint-based programming, specific document grounding, iterative methods, example-based formats, and custom tagging for specific cases. **Instruction Misinterpretation:** Example Prompt Instruction: Keep the passage brief. Here, the user intent of what brief means could be different from what the model interprets. This commonly occurs in cases of subjective instructions.

Instruction Overlooking: Here, the model might miss a certain instruction within a complex prompt

Inconsistent Response Generation: Here, the model but generate different responses and portray varying response patterns in every response generation for the same prompt.

Hallucinations: These are cases when the model produces untrue information or content that the user did not ask for.

Incomplete generation: These are cases of the model not completing generation consistently.

4 EXPERIMENT ON CRITERIA BASED EVALUATION VS BASELINE

Experiment Setup: To evaluate our criteria-based question-generation approach, we conducted an experiment comparing it with a baseline method. The baseline used the following single-prompt template:

"Evaluate how well the following output aligns with the given prompt. Check if each instruction is addressed in the output. Return alignment scores for each instruction and an aggregate alignment score."

We selected 10 prompts from the Dolomites dataset [41], each containing an average of 10 instructions. Outputs for each prompt were generated using OpenAI's GPT-4o model. We then compared the alignment results produced by our criteria-based evaluation approach against those of the baseline method.

To assess alignment coverage, we evaluated whether all user instructions were addressed in the alignment reports. This evaluation was conducted by measuring entailment between the alignment report and the user instructions in the form of "entailment" or "contradiction". We used the popularly used NLI-RoBERTa entailment model (cross-encoder/nli-roberta-base) for this experiment.

Results: Our criteria-based approach achieved 100% coverage of user instructions by first generating criteria questions from the guidelines and then systematically evaluating prompt outputs against each criterion. This structured approach ensures that no instruction is overlooked.

In contrast, the baseline approach achieved only 76.5% coverage, indicating that the single-prompt method may fail to address some user instructions. These results demonstrate that breaking down instructions into individual criteria leads to a more thorough and accurate evaluation.

User Study Validation: Further validation was conducted through user study log analysis, which revealed that participants made edits to only 17% of the automatically generated criteria. This low edit

rate suggests that users found the generated criteria to be both accurate and useful in assessing their requirements.

Our findings highlight the effectiveness of a granular, criteria-based evaluation approach in ensuring comprehensive coverage of user instructions. While the single-prompt method offers simplicity, it risks incomplete assessment. The low rate of edits from users further supports the accuracy and reliability of our criteria-generation method.

5 USER INTERFACE

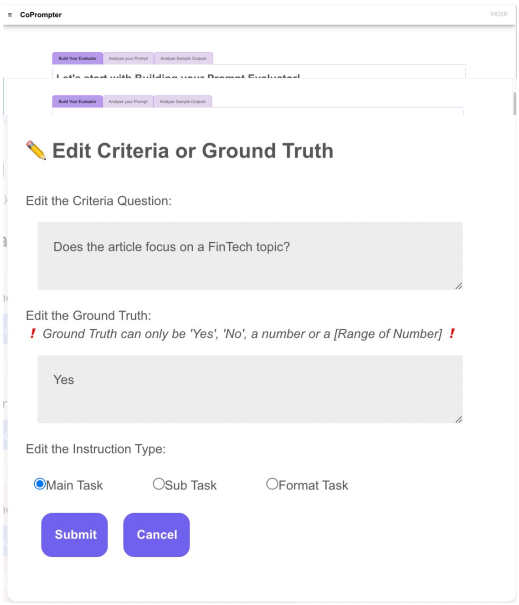

Figure 12: Edit criteria

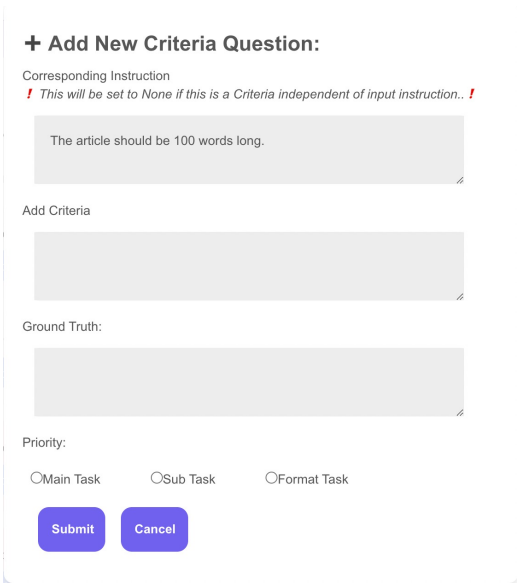

Figure 13: Add criteria

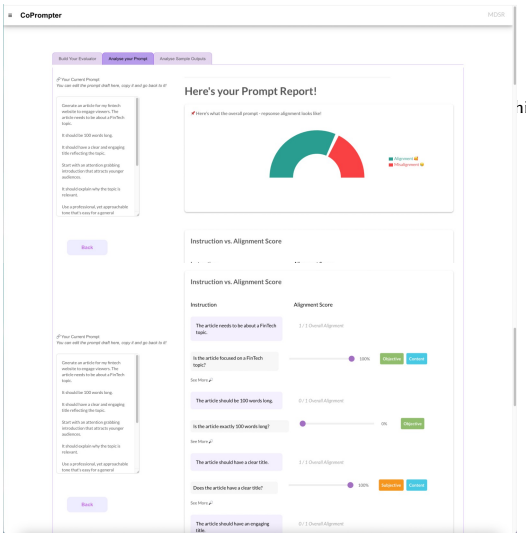

Figure 15: Instruction vs alignment score

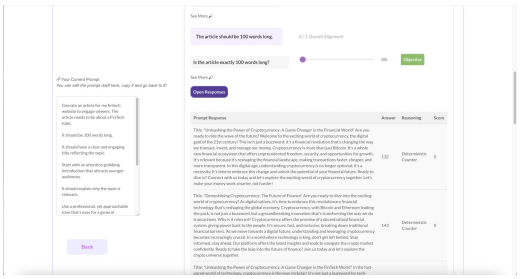

Figure 16: Per-criterion report

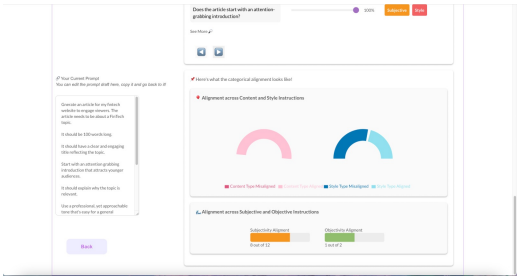

Figure 17: Categorical Alignment

6 USER STUDY TASK

User Task

You are building a website for your <fintech> product (you may choose the domain expertise) to engage potential customers. A feature of the website randomly generates articles in FinTech for the visitors to enhance engagement. You have to craft a prompt that performs this task. Below are some guidelines that can be used to write a good article. You are free to update them as you like! You can add more specifications, constraints and formats. This is your app!

7 SYSTEM USABILITY SCALE (SUS) QUESTIONS

S.No	Question
1	I think that I would like to use this system frequently (for the intended use case) (System Usability).
2	I found the system unnecessarily complex. (System Efficiency).
3	I thought the system was easy to use. (System Efficiency).
4	I think that I would need the support of a technical person to be able to use this system. (System Efficiency).
5	I found the various functions in this system were well integrated. (System Usability).
6	I thought there was too much inconsistency in this system. (System Usability).
7	I would imagine that most people would learn to use this system very quickly. (System Efficiency).
8	I found the system very cumbersome to use. (System Efficiency).
9	I felt very confident using the system. (System Usability).
10	I needed to learn a lot of things before I could get going with this system. (System Usability).

Table 1: System Usability and Efficiency Questions

8 PROMPTS

The core of CoPrompter is a set of prompting strategies. In this section, we list down each prompt used for CoPrompter, its sample input and output.

Task Objective Generation

I. Prompt

Given a task prompt between `<Task prompt>` and `</Task prompt>` provided by the user, your goal is to generate a task objective that concisely captures the overall goal of the task.

Guidelines:

(1) The task objective must convey the main purpose of the prompt in fewer than 10 words.
(2) Do not include specific details or subjective terms from the task prompt.
(3) Focus solely on the overall goal of the task without adding any new information or interpretations.
(4) The objective should be clear, objective, and concise, avoiding any subjectivity or unnecessary elaboration.
(5) For example, if the task prompt is:

"Generate a non-disclosure agreement of two pages (each page is limited to 250 words) for a software development project involving Party A and Party B. The confidentiality duration should be 5 years. The first page should include definitions for key terms such as 'confidential information', 'disclosure', and 'recipient'. On the second page, provide clauses detailing the protocol for the return or destruction of confidential information, exceptions to maintaining confidentiality, and the repercussions following a breach of the agreement. Please indicate the separation between the first and second pages with a full line of dashed lines ('——'). Also, make sure that each page is clearly labeled with its respective page number."

The task objective would be:

```
"Generate non-disclosure agreement for a
software development project."
```

<Task prompt>
task prompt
</Task prompt>

Output:

```
Task objective = [] (python list)
```

II. Sample Input/Output:

Input:

> The article needs to be about a Credit Default Swaps. It should be 100 words long. It should have a clear and engaging title reflecting the topic. Start with an attention grabbing introduction. It should explain why the topic is relevant. Use a professional, yet approachable tone that's easy for a general audience to understand. Avoid excessive jargon. You are allowed to use GenZ slang. Make sure the article generated is not hurtful, biased against any community/person. There should be obvious takeaways for readers in the article. End with a strong CTA that encourages readers to engage with the company. It should not be boring to read. It

is necessary that the information provided is factually accurate. Don't hallucinate extra details.

Output:

"Write a 100-word article about Credit Default Swaps."

Atomic Instruction Generation

I. Prompt

You are an intelligent assistant tasked with decomposing guidelines into the most granular sub-instructions possible.**Guidelines:**

(1) **Decomposition:** Break down each instruction in the guidelines until it cannot be further decomposed.
- A sub-instruction should contain only one singular instruction.
- Decompose any comma-separated instructions into individual sub-instructions.
- For example, the instruction "Write a brief, friendly introduction" should be broken down into:
  (a) Write an introduction.
  (b) The introduction should be brief.
  (c) The introduction should be friendly.

(2) **Exact Wording:** Use the exact wording from the guidelines when creating sub-instructions. Do not rephrase or add new instructions.

(3) **Exclusion:** Do not include lines that are not instructions (e.g., headers, descriptions).

Output Format: Each sub-instruction should be formatted exactly as follows, with no deviation:

- `#### Atomic Instruction: [The most granular form of the instruction.]`
- `- Corresponding Instruction in the Guidelines: [The original instruction from which the atomic instruction was derived.]`

Ensure the following:

- Each sub-instruction starts with `#### Atomic Instruction:`.
- Each corresponding instruction starts with `- Corresponding Instruction in the Guidelines:`.
- There are no additional bullet points, numbers, or formatting changes.

Task Prompt: task prompt

II. Sample Input/Output:

Input: Same input as the previous prompt (Task Objective Generation).
Output: We show some atomic instructions and corresponding guidelines instructions for the input.

1 – Atomic Instruction: "The article needs to be about a Credit Default Swaps."
  – Corresponding Instruction in the Guidelines: "The article needs to be about a Credit Default Swaps.'
2 – Atomic Instruction: "Use a professional tone."
  – Corresponding Instruction in the Guidelines: "Use a professional, yet approachable tone that's easy for a general audience to understand."
3 – Atomic Instruction: "Use a approachable tone."
  – Corresponding Instruction in the Guidelines: "Use a professional, yet approachable tone that's easy for a general audience to understand."
4 – Atomic Instruction: "The tone should be easy for a general audience to understand."
  – Corresponding Instruction in the Guidelines: "The tone should be easy for a general audience to understand."

Evaluation Criteria Generation

Prompt for Evaluation Criteria Generation

You will be given guidelines for a task about `"task_objective"`. You will also receive a list of specific sub-instructions extracted from the guidelines. Your task is to generate concise evaluation questions that assess whether the response generated by another LLM adheres to each sub-instruction. Additionally, assign a priority level to each question based on the importance of the sub-instruction. **Guidelines:**

(1) **Direct Evaluation:** For each sub-instruction, create an evaluation question that directly assesses whether the response effectively follows the sub-instruction. Avoid simply rephrasing the sub-instruction as a question; instead, capture the intended outcome clearly.
(2) **Contextual Understanding:** Use the context provided by the corresponding prompt instruction to guide your understanding of the sub-instruction's intent.
(3) **Text-Based Focus:** Ensure all evaluation questions are focused solely on assessing text responses, without consideration for visual content like images or diagrams.
(4) **Simplicity and Clarity:** Formulate questions using clear, straightforward language to make them easily understandable.
(5) **Priority Levels:** Assign a priority level to each evaluation question based on its importance:
- **Level 1:** Most important, for critical sub-instructions essential to the task.
- **Level 2:** Important but secondary, for more detailed or supplementary instructions.
- **Level 3:** Formatting and presentation-focused, not essential to the core logic of the task.

Output Format:

- Each sub-instruction is labeled `"Sub-Instruction X:"`, where X represents the sub-instruction number.
- The corresponding evaluation question starts with `"Evaluation Question:"`.
- The priority level is labeled `"Priority:"`, with the assigned level.

Example Input: Sub-Instruction 1: "Describe the methodology in detail." | Corresponding Prompt Instruction: "Provide a detailed description of the methodology used in the research, including data collection and analysis methods."
Sub-Instruction 2: "Ensure the methodology section is clear and easy to understand." | Corresponding Prompt Instruction: "The methodology section should be written clearly, avoiding technical jargon that could confuse readers unfamiliar with the topic."
Sub-Instruction 3: "Use charts or diagrams to illustrate the methodology where possible." | Corresponding Prompt Instruction: "To enhance clarity, use charts or diagrams to illustrate the methodology whenever possible."

Example Output: Sub-Instruction 1: Describe the methodology in detail. Evaluation Question: Does the response provide a detailed description of the methodology, including data collection and analysis methods? Priority: Level 1
Sub-Instruction 2: Ensure the methodology section is clear and easy to understand. Evaluation Question: Is the methodology section clear and free of technical jargon that could confuse readers? Priority: Level 2
Sub-Instruction 3: Use charts or diagrams to illustrate the methodology where possible. Evaluation Question: Are charts or diagrams used effectively to illustrate the methodology in the text? (Focus on how they are described rather than their visual content.) Priority: Level 3

II. Sample Input/Output:

Input: Input for this prompt is the output from previous atomic instruction step. A sample format is provided in the prompt itself.
Output: For the atomic instructions provided in previous section, we get the following criteria:

1 'Does the article focus on Credit Default Swaps as the main topic?'
2 'Is the tone of the article professional?'
3 'Is the tone of the article approachable and easy to understand for a general audience?'
4 'Is the article written in a way that is easy for a general audience to understand?'

Criteria Metadata Generation

I. Prompt

You are an AI assistant tasked with analyzing evaluation questions for subjectivity and generating metadata to aid in evaluation. Your goal is to provide a comprehensive analysis of the given evaluation question, considering its context within the task objective, complete instruction, and atomic instruction.

Provided Components: 1. **Task Objective:** Enclosed in `<TASK>` tags. 2. **Complete Instruction:** Enclosed in `<COMPLETE INSTRUCTION>` tags. 3. **Atomic Instruction:** Enclosed in `<ATOMIC INSTRUCTION>` tags, extracted from the complete instruction. 4. **Evaluation Question:** Enclosed in `<EVALUATION QUESTION>` tags, used to assess LLM output compliance with the prompt instruction.

Components for Analysis: `<TASK>` task objective `</TASK>` `<COMPLETE INSTRUCTION>` prompt instruction `</COMPLETE INSTRUCTION>` `<ATOMIC INSTRUCTION>` atomic instruction `</ATOMIC INSTRUCTION>` `<EVALUATION QUESTION>` base criteria `</EVALUATION QUESTION>`

Subjectivity Analysis: Evaluate if the question is subjective. A question is subjective if it allows multiple interpretations or requires additional context to be understood objectively. If subjective: 1. Identify subjective terms/phrases. 2. Provide objective interpretations. 3. Offer good and bad examples for each interpretation. 4. Assign a similarity score (1-5) with rationale.

Metadata Determination: 1. **Evaluation Type:** 'Basic LLM' (qualitative) or 'Count LLM' (quantitative). 2. **Question Theme:** 'Content', 'Style', or 'Format'. 3. **External Input Required:** 'Yes' or 'No'. 4. **Ground Truth Answer:** 'Yes', 'No', number, or range (e.g., 200-250).

Output Format: `<metadata>`
Type: [Explicit or Implicit]
Subjectivity Present: [Yes or No]
Subjective Term/Phrase: "[If applicable]"
Interpretations: - Interpretation 1: "[If applicable]" - Good Example 1: "[If applicable]" - Bad Example 1: "[If applicable]" - Interpretation 2: "[If applicable]" - Good Example 2: "[If applicable]" - Bad Example 2: "[If applicable]"
Similarity Score with Reason: "[If applicable]"
Evaluation Type: [Basic LLM or Count LLM]
Question Theme: [Content or Style or Format]
External Input Required: [Yes or No]
Ground Truth Answer: [(Yes or No) or (Number) or (Range of Numbers)] `</metadata>`

Note: Ensure the response follows the specified format without additional text or explanations.

II. Sample Input/Output:

Input: Input for this prompt is the output from previous criteria generation step.
Output: For the criteria provided in previous section, we get the following metadata:

1 'Does the article focus on Credit Default Swaps as the main topic?'
   – Type: Explicit
   – Subjectivity Present: No
   – Subjective Term/Phrase: N/A
   – Similarity Score with Reason: N/A
   – Evaluation Type: Basic LLM
   – Question Theme: Content
   – External Input Required: False
   – Ground Truth Answer: Yes
2 'Is the tone of the article professional?'
   – Type: Explicit
   – Subjectivity Present: Yes
   – Subjective Term/Phrase: 'professional'
   – Interpretation 1: Adheres to industry standards and avoids casual language
      ∗ Good Example: Credit Default Swaps (CDS) are financial derivatives that function as a type of insurance against the default of a borrower.
      ∗ Bad Example: Credit Default Swaps are kinda like insurance for loans, you know?
   – Interpretation 2: Uses formal language and maintains a serious tone
      ∗ Good Example: Credit Default Swaps are sophisticated financial instruments used to hedge against credit risk.
      ∗ Bad Example: Credit Default Swaps are cool tools for managing credit risk.
   – Similarity Score with Reason: N/A
   – Evaluation Type: Basic LLM
   – Question Theme: Style
   – External Input Required: False
   – Ground Truth Answer: Yes

Descriptive Criteria Evaluation

I. Prompt

Descriptive Criteria Prompt:
For the given input text below, answer the following question in yes or no format. Also, provide your reasoning for the answer.
Input Text: `{llm_output}`
QUESTION: `{criteria_question}`
The output will be in this format: `{{'answer': '<yes/no here>', 'reasoning': '<reason here>'}}`

II. Sample Input/Output:
Input:

- Criteria: "Is the tone of the article professional?"
- Prompt output: "Demystifying Credit Default Swaps: A Simple Guide (Entire article here)"

Output:

- Answer: Yes
- Reasoning: The introduction explains that Credit Default Swaps (CDS) are financial instruments that act like insurance policies for bonds, providing protection against default. They are relevant because they play a significant role in global finance, affecting everything from individual investments to the health of entire economies.

Layered Criteria Evaluation

I. Prompt

Layered Measurable Criteria Prompt:
For the given input text below, perform the following steps:

1. Analyze the question to determine what needs to be counted (e.g., words, sentences, paragraphs, questions, or other elements).
2. Identify the portion of the text that is relevant to the question.
3. Count the relevant elements based on your analysis of the question.
4. Provide the count, the type of count (e.g., 'word', 'sentence', 'question', etc.), the portion of the text you analyzed, and explain how you determined what to count and which portion of the text was analyzed.

Input Text: `{llm_output}`
QUESTION: `{criteria_question}`

The output will be in this format:

```
{{'count_type': '<count type here>',
'answer': '<count here>',
'feature_text': '<portion of the text analyzed>',
'reasoning': '<explanation here>'}}
```

II. Sample Input/Output:
Input:

- Criteria: "Is the article exactly 100 words long?"
- Prompt output: "Demystifying Credit Default Swaps: A Simple Guide (Entire article here)"

Output:

- Count Type: "word"
- Answer: 160
- Feature Text: Same article.
- Reasoning: Deterministic Counter.