EvalLM: Interactive Evaluation of Large Language Model Prompts on User-Defined Criteria

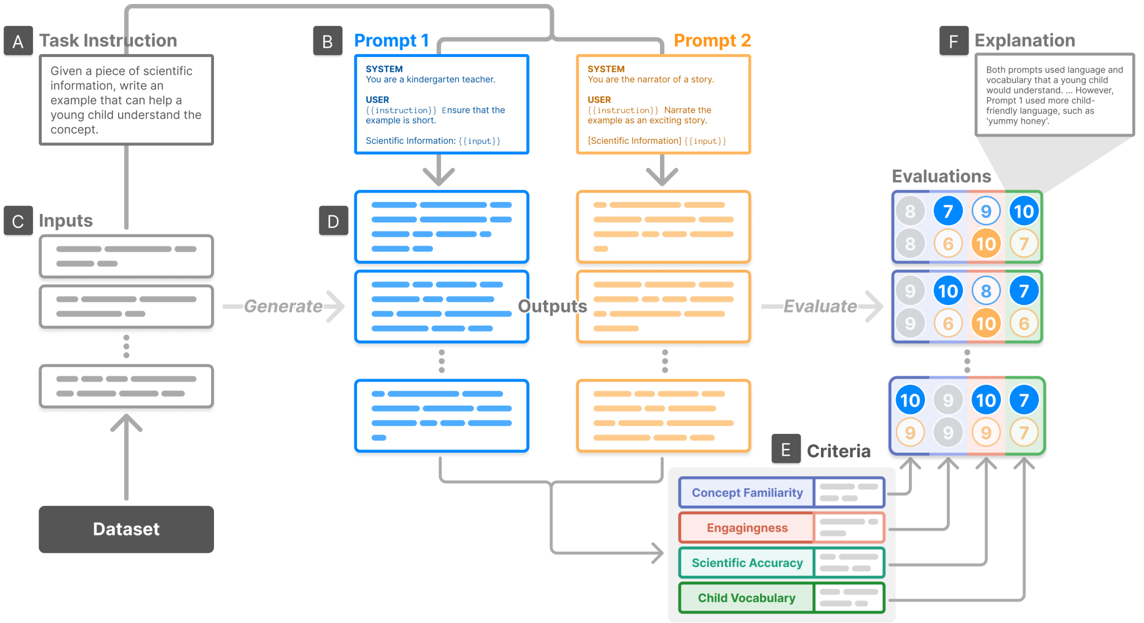
Tae Soo Kim
taesoo.kim@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Yoonjoo Lee
yoonjoo.lee@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

Jamin Shin
jayshin.nlp@gmail.com
NAVER AI Lab
Seongnam, Republic of Korea

Young-Ho Kim
yghokim@younghokim.net
NAVER AI Lab
Seongnam, Republic of Korea

Juho Kim
juhokim@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea



Given a piece of scientific information, write an example that can help a young child understand the concept.

SYSTEM
You are a kindergarten teacher.

USER
{{instruction}} Ensure that the example is short.

Scientific Information: {{input}}

SYSTEM
You are the narrator of a story.

USER
{{instruction}} Narrate the example as an exciting story.

[Scientific Information] {{input}}

Both prompts used language and vocabulary that a young child would understand. ... However, Prompt 1 used more child-friendly language, such as 'yummy honey'.

Figure 1: EvalLM aims to support prompt designers in refining their prompts via comparative evaluation of alternatives on user-defined criteria to verify performance and identify areas of improvement. In EvalLM, designers compose an overall task instruction (A) and a pair of alternative prompts (B), which they use to generate outputs (D) with inputs sampled from a dataset (C). Then, based on the criteria that the user defined (E), the system automatically evaluates these outputs to compare how each prompt performed on each criterion and provides explanations to support the user's verification of these explanations (F).

ABSTRACT

By simply composing prompts, developers can prototype novel generative applications with Large Language Models (LLMs). To refine prototypes into products, however, developers must iteratively revise prompts by evaluating outputs to diagnose weaknesses. Formative interviews (N=8) revealed that developers invest significant effort in manually evaluating outputs as they assess context-specific and subjective criteria. We present EvalLM, an interactive system for iteratively refining prompts by evaluating multiple outputs on user-defined criteria. By describing criteria in natural language, users can employ the system's LLM-based evaluator to get an overview of where prompts excel or fail, and improve these based on the evaluator's feedback. A comparative study (N=12) showed that EvalLM, when compared to manual evaluation, helped participants compose more diverse criteria, examine twice as many outputs, and reach satisfactory prompts with 59% fewer revisions. Beyond prompts, our work can be extended to augment model evaluation and alignment in specific application contexts.

CHI '24, May 11–16, 2024, Honolulu, HI, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0330-0/24/05
https://doi.org/10.1145/3613904.3642216

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**; *Empirical studies in HCI*; • **Computing methodologies → Natural language processing**.

KEYWORDS

Large Language Models, Natural Language Generation, Evaluation, Human-AI Interaction

ACM Reference Format:
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 (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3613904.3642216

1 INTRODUCTION

Large Language Models (LLMs) have catalyzed the creation of a wide array of novel applications. Composed of billions of parameters and trained on billions of tokens, LLMs can interpret a natural language description of a task, a **prompt**, and generate coherent human-like outputs for diverse purposes [9, 39, 46] (e.g., summarization [69], dialogue [63], story writing [13]). By composing a prompt, developers and researchers (i.e., prompt designers) can guide LLMs to perform novel tasks that satisfy desired requirements and support specific application settings. For example, HCI researchers have leveraged the generative capabilities of LLMs to ideate possible journalistic angles for a given event [50], generate questions to quiz children about information they learned [33], or simplify research papers into plain language [4].

Although prompt designers can easily bootstrap AI-based applications by simply composing a prompt, developing a prototype into a polished application that consistently produces high-quality outputs requires more dedicated effort. As LLMs are non-deterministic and even partial changes in a prompt can significantly influence generated outputs [37, 41], designers need to iterate on their prompts multiple times to achieve satisfactory results [27, 39, 57, 69, 72, 73]. In this iterative process, designers test their prompt with sample inputs (e.g., paragraphs to summarize), inspect the generated outputs to identify areas for improvement, revise their prompts (e.g., change structure, wording, content), and repeat. When designers adopt LLMs for more open-ended generative tasks, however, evaluating outputs becomes significantly more challenging as no automatic metrics can adequately encode and measure the subjective quality of outputs [14]. Due to the lack of suitable automatic metrics, generative tasks are typically evaluated by human annotators or experts [23], but these can be impractical during early development stages when designers need to quickly iterate on prompts.

To understand how evaluation challenges affect the development of LLM-based applications, we conducted formative interviews with 8 prompt designers (e.g., developers, and researchers in HCI and ML) to understand how they iterate on and evaluate their prompts. Our interviews revealed that designers considered multiple *criteria* that were unique and specific to their applications when evaluating outputs from their prompts. Due to the novelty of these criteria and the significant cost of recruiting annotators, however, designers had to manually evaluate their prompt outputs themselves. As this manual and multi-faceted evaluation of outputs incurred a significant cognitive load, designers could only evaluate small batches of outputs and only on a subset of their criteria. As a result, when they refined their prompts, designers could not fully verify how their refinements had affected output quality or identify where further refinements were needed.

Based on these findings, we introduce EvalLM to facilitate prompt iterations by supporting the evaluation of outputs on user-defined and application-specific criteria (e.g., measuring "Object Familiarity" in scientific analogies for children). Instead of focusing on the low-level task of assessing generated outputs, EvalLM shifts designers' focus to the higher-level process of refining prompts and criteria—representations of their plans and requirements. Inspired by recent techniques for LLM-based evaluations [40, 71, 76], EvalLM employs an LLM as both (1) an *evaluation assistant*, which evaluates outputs on the defined criteria, and (2) a *criteria reviewer*, which revises the defined criteria. To aid users in revising their prompts and criteria, the evaluation assistant explains its assessments, allowing the user to identify where prompt outputs fell short or to identify where the assistant's interpretation of criteria misaligned with their own. Furthermore, the criteria reviewer analyses the user's criteria to identify revisions that can lead to evaluations of outputs on more specific and fine-grained dimensions. Through iterations of this collaborative process, designers co-evolve their prompts and criteria, where prompts improve to satisfy criteria and criteria improve to discern the quality of prompts—ultimately leading to more polished applications.

To understand how prompt designers adopt automatic evaluations during prompt iterations, we conducted a within-subjects study (N=12) where participants improved and evaluated prompts for novel tasks proposed by recent HCI work. In the study, participants used both EvalLM and a baseline where they manually evaluated outputs—emulating designers' current practice. Our study revealed that EvalLM helped participants "debug" their prompts by allowing them to quickly identify areas for improvement, and the evaluation assistant's explanations served as feedback by helping participants think about how to make these improvements. As a result, we observed that participants reached satisfactory prompts more efficiently as they tested 59% fewer changes than when they did not have evaluation assistance. As EvalLM also facilitated criteria revision, participants felt higher satisfaction regarding the quality of their criteria—suggesting that these criteria could be valuable during human evaluations. Overall, these findings suggest that EvalLM can fill the current gap between application development and deployment by assisting designers to iterate on prompts until a stage where they have the confidence to commit resources for more robust human evaluations.

This work presents the following contributions:

(1) Qualitative findings from interviews with prompt designers ($N = 8$) that revealed how the effort of manually evaluating outputs on multiple, task-specific criteria can inhibit designers from making informed decisions during the iteration process.
(2) EvalLM, an interactive system that aids users in revising prompts and verifying the effect of revisions by employing

an LLM-based evaluation assistant to assess outputs on user-defined criteria, and a criteria reviewer to refine these criteria to assess more specific and detailed dimensions of outputs.

(3) Findings from a user study ($N = 12$) that demonstrated how EvalLM can aid designers in debugging their prompts and ideating on strategies to more effectively revise their prompts.

2 RELATED WORK

This work aims to support the iteration of LLM prompts for novel generative tasks by supporting interactive evaluation of outputs. To understand this space, we review literature in (1) prompt design challenges and related support, (2) natural language generation, and (3) interactive evaluation in broader machine learning.

2.1 Designing LLM Prompts

Although LLMs facilitate the development of novel AI-based applications without the need for data collection or model training, designing satisfactory prompts can be an arduous task [39]. The specific format, phrasing, content, examples, or even the order of examples used in prompts can significantly affect performance [37, 39, 41, 55, 75]. However, as the space of possible natural language instructions is near infinite, designers need to test as many possibilities as possible to identify high-performing prompts [38, 72]. To help designers identify effective prompts, researchers have proposed various tools that facilitate prompt design. For example, *AI Chains* [69] helps users decompose complex tasks into a chain of prompts that can be tested individually, and Kim et al. [30] proposed a design framework for interfaces that support end-users to testing and experimentation with prompts. To automate testing, various systems [44, 57, 66] allow users to create prompt variants and automatically compare their performance. However, while these prior approaches focused on classification tasks, evaluating performance in open-ended generative tasks can be more complex as outputs need to be examined on subjective criteria, which typically requires human effort. Building on these systems, our work proposes a human-AI collaborative system where users define subjective evaluation criteria and an LLM automatically assesses outputs on these criteria to provide insight into the performance of prompts and surface necessary improvements.

2.2 Natural Language Generation and Evaluation

Natural language generation (NLG) is the family of NLP tasks where the goal is to generate text that satisfies a communicative goal (e.g., summarize a document) while possessing several desired qualities (e.g., fluent, coherent) [20, 23, 24, 45]. While recent years have brought significant progress in NLG, especially due to LLMs, a constant obstruction to progress has been the difficulty of evaluating these tasks [17, 28, 70]. Unlike classification tasks where performance is measured by comparing a prediction to a ground-truth label, generation tasks are *ill-posed*—i.e., multiple dissimilar outputs can be equally valid. While researchers have proposed automatic metrics that compare outputs to several ground-truth references (e.g., *BLEU* [48], *ROUGE* [36]), the space of valid outputs in open-ended generative tasks can be overwhelmingly vast, making it nearly impossible to create sufficiently comprehensive references sets. Thus, human evaluations where annotators rate or rank generated text have become the golden standard [23]. However, the cost and effort involved in recruiting human annotators can make this type of evaluation prohibitive during early development stages. As an alternative, recent work [12, 22, 35, 40, 71, 76] has employed LLMs to simulate annotators and automatically evaluate outputs on their overall quality or a pre-defined set of criteria—demonstrating agreement with human evaluations on par with the level of agreement between human evaluators [76]. Inspired by these approaches, we incorporate LLM-powered simulated evaluators to support interactive evaluation of LLM prompts on user-defined criteria and investigate how users interact with these evaluations to refine prompts.

2.3 Interactive Machine Learning Evaluation

Evaluation is fundamental to the development of machine learning models for real-world applications. Beyond assessing performance on a single metric, practitioners (e.g., developers, researchers, engineers) assess more fine-grained model behaviors to identify flaws and potential improvements [52]. To support fine-grained assessments, prior work has introduced various systems that allow practitioners to interactively evaluate outputs. For example, *Zeno* [11], the *What-If Tool* [65], and *Errudite* [67] help practitioners to identify *slices* or subsets of data that may reveal distinct model failures. Beyond testing on existing input data, *Polyjuice* [68] and *AdaTest* [51] allow practitioners to generate potentially challenging input data to test a model's behavior. To aid practitioners in resolving issues after models are deployed, *Angler* [53] combines online and offline data to help practitioners prioritize performance issues, and *Deblinder* [10] allows practitioners to collect and analyze model failure reports from crowdworkers. Our work expands on these ideas by allowing practitioners to interactively evaluate prompt outputs on fine-grained criteria with the aid of an LLM-based evaluator, which could simulate the feedback provided by end-users.

3 FORMATIVE INTERVIEWS

To understand current practices and challenges when evaluating and iterating on LLM prompts, we conducted interviews with prompt designers. These interviews focused on understanding how prompt designers evaluate performance during early development stages and how these evaluations inform their refinement of prompts.

3.1 Participants and Procedure

We recruited 8 prompt designers through posts on online forums within our institution and word-of-mouth. These participants held various roles related to generative applications, and came from both academia and industry: 2 graduate students in HCI (1 MS, 1 PhD), 2 MS students in ML/NLP, 2 research scientists at a large company, 1 data scientist at a startup, and 1 startup CEO. All of the participants mentioned working on at least one project where they designed prompts for a novel generative applications. Their applications covered diverse contexts: social media, document question-answering, image captioning, writing, teaching, and intelligent assistants.[1]

[1] These are described broadly to preserve confidentiality.

The length of their experiences with intensive prompt engineering ranged from 4 months to more than 1 year. Participants were compensated with approximately 45 USD (60,000 KRW) for the 1-hour interview. The interviews were conducted in a semi-structured format, and were recorded, manually transcribed and coded through a thematic analysis.

3.2 Findings

All of the designers mentioned working on applications for which they defined novel generation tasks. These tasks were novel as they (1) introduced new requirements to pre-existing generation tasks, or (2) were not analogous to any pre-existing tasks, according to participants. To develop the prompts for these applications, all of the designers first composed an initial prompt based on a preliminary set of expectations and requirements and, then, iteratively evaluated and refined their prompt to guide the LLM to better meet these expectations.

3.2.1 Evaluation is Manual. All of the designers mentioned how, at each iteration step, they tested their prompts on sample inputs and then manually evaluated the outputs themselves. Designers mentioned that they had to evaluate manually since they considered aspects that were subjective (P1-3, P5-8) and specific to their task (P1-8), meaning that there were no existing automated metrics to measure these aspects. Furthermore, since they were still in the development stage, recruiting annotators would be prohibitively expensive and would slow down the iteration process (P1-2, P4). However, all of the designers mentioned how manual evaluation could be demanding and time-consuming and, thus, they only tested prompts on a small set of sample inputs (i.e., one to three samples) at each iteration step—which was still taxing especially with lengthier outputs (P1-2).

3.2.2 Evaluation is Multi-Faceted. Due to the complexity of the designers' intended applications, performance or the quality of the outputs could not be determined with a single criterion. Instead, designers considered multiple criteria or factors simultaneously when examining outputs, which made evaluation significantly more challenging (P1, P3-5). P5 mentioned how they had to carefully examine the outputs to *"catch all the subtleties"*. As this involved significant cognitive load and effort, designers described various ways in which they handled this multi-faceted evaluation. For example, four designers (P2, P4-5, P7) said that they only focused on the most important criteria for their task, and two others (P5-6) simplified assessments to assigning binary ratings on each criterion—i.e., whether the criterion was satisfied or not. Alternatively, P4 resorted to evaluating and refining one criterion at a time, but P7 noted that this method did not work for them as refining the prompt for one criterion led to the prompt failing at previously *"resolved"* criteria. Thus, while designers ideally wanted to evaluate holistically on multiple criteria, the effort required could lead them to only partially evaluate outputs.

3.2.3 Evaluation is Dynamic. Several designers described how they started the prompt design process with an initial set of evaluation criteria based on the intended goals for their application (P1-3, P5) and prior work (P2-5). Additionally, designers also expanded and transformed their criteria in each prompt iteration. By examining outputs, they identified additional criteria to consider as they observed flaws in the outputs, which they had previously not expected, or because they recognized other aspects that they wanted to improve on (P1-2, P4, P7-8). Beyond adding criteria, designers also mentioned how they had to concretize their criteria by determining how they should be evaluated. However, as these criteria could be subjective, it could be challenging to define what "success" meant for each criterion (P4-7). For designers who worked in teams, they mentioned how they would concretize the criteria by discussing with team members who could provide different perspectives (P4-5).

3.2.4 Evaluation to Refinements. Through the evaluations, designers identified what criteria the generated outputs failed to satisfy, and they attempted to refine their prompts to improve on these dimensions. However, most designers (P1-7) mentioned how they were unsure about how they should revise their prompts—the well-known challenge of prompt engineering [72]. Designers mentioned how they had no alternative, but to simply test different changes and to manually evaluate outputs again to check the effect of these changes. As this involved significant effort, designers mentioned that they could struggle to verify how much a revision improved quality on specific criteria (P3-4, P6). Due to the overall complexity of the evaluation-refinement process, designers also mentioned how they would fail to record all of their prompt revisions and their effects, which prevented them from learning from previous attempts and tracking their progress (P4-5, P7).

4 DESIGN GOALS

To support prompt iteration, we must support efficient evaluation of outputs on designers' own criteria. Recent work [40, 71] showed that LLMs can evaluate text on diverse subjective criteria—revealing their potential as evaluation assistants. However, for designers to effectively use LLMs as evaluators, they must be able to define their own criteria and verify that subsequent evaluations align with their expectations. To scaffold these interrelated processes, we distill insights from our interviews into design goals for an LLM-based prompt iteration and evaluation system. We additionally take high-level inspiration from the process for developing psychometric scales [54]—question sets that collectively measure a variable (e.g., behavior, feeling) that cannot be assessed directly (e.g., NASA-TLX [25])—as we notice parallels between scale questions and evaluation criteria.

- **DG1: Automate evaluation of generated outputs according to user-defined criteria.** An automatic evaluation assistant can reduce effort by providing an initial assessment of outputs that designers can then verify. By defining their own criteria, designers can align the assistant's assessments with their own expectations and requirements.
- **DG2: Facilitate inspection of automatic evaluations through explanations.** Similar to how cognitive interviews can reveal incorrect or unclear questions in scales by asking respondents to verbalize their thoughts [7], the automatic evaluator should explain and justify its evaluations so that designers can inspect whether the evaluations align with their expectations.

- **DG3: Allow for the definition of criteria based on output data and prior literature.** As revealed by our interviews, prompt designers envision new criteria by assessing outputs and also by referring to prior work—resembling the inductive and deductive methods for defining psychometric scale questions [7].
- **DG4: Review the user-defined criteria to identify potential revisions.** Inspired by how scales are revised through reviews from external judges [7, 54], a system should aid designers in reviewing criteria to identify potential revisions, which could increase the effectiveness of subsequent evaluations.
- **DG5: Surface unreliable evaluations during large-scale evaluations.** While designers need to evaluate larger samples to comprehensively assess performance, they may be unable to verify all these evaluations. As an alternative, we take inspiration from reliability tests (e.g., inter-rater, test-retest) for psychometric scales [7, 54] and suggest that an evaluation system should surface less reliable evaluations for designers to verify.
- **DG6: Aid designers in tracking and comparing the effect of prompt changes.** As stated in the interviews, it can be challenging to understand the effect of each prompt change. By helping designers track how changes affect performance in the automatic evaluations, designers can make more informed iteration decisions.

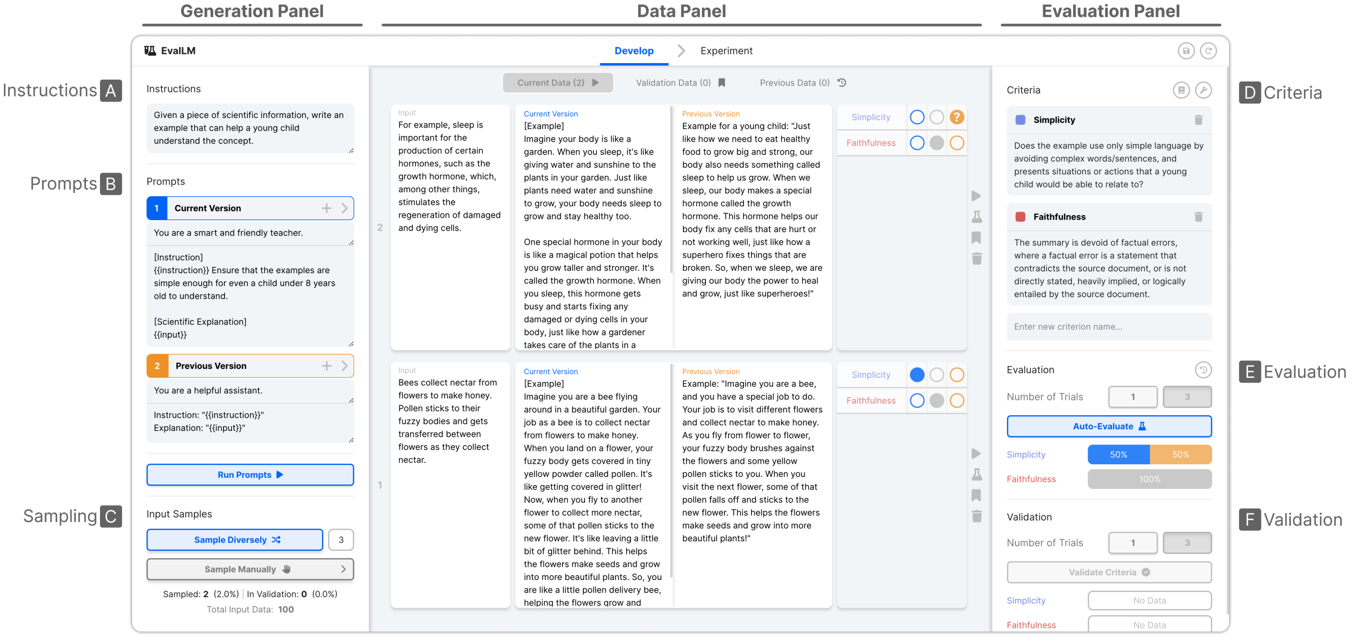
5 EVALLM

Based on these design goals, we present *EvalLM* (Fig. 2), an interactive system for iterating on prompts by evaluating multiple outputs on multiple user-defined criteria. While designers typically compose prompts and assess outputs to provide feedback to themselves, EvalLM transforms this into a collaborative process where a designer iteratively refines prompts and criteria based on feedback from an LLM (DG1). Specifically, the system employs an LLM as both an *evaluation assistant* and *criteria reviewer*. The evaluation assistant judges prompt outputs based on the user's definitions of criteria and explains its evaluations, which can reveal where the prompts fall short or where criteria may be unclear (DG2). The user can define and revise criteria at any time (DG3) and, to facilitate this, the criteria reviewer recommends potential revisions that can enhance the detail and specificity of evaluations (DG4).

5.1 Interface

To illustrate the interactions in EvalLM, we walk through an example of an ML practitioner, Emily, who is designing prompts for a novel task of generating examples that help young children understand complex scientific phenomena.

5.1.1 Composing Prompts. In the generation panel, the user writes the overall instruction for their task (Fig. 2A) and designs two prompt templates (Fig. 2B). EvalLM is designed for comparing prompts as this enables designers to compare the performance of



Given a piece of scientific information, write an example that can help a young child understand the concept.

You are a smart and friendly teacher.

[Instruction]
{{instruction}} Ensure that the examples are simple enough for even a child under 8 years old to understand.

[Scientific Explanation]
{{input}}

You are a helpful assistant.

Instruction: "{{instruction}}"
Explanation: "{{input}}"

Sampled: 2 (2.0%) | In Validation: 0 (0.0%)
Total Input Data: 100

Input
For example, sleep is important for the production of certain hormones, such as the growth hormone, which, among other things, stimulates the regeneration of damaged and dying cells.

Current Version
[Example]
Imagine your body is like a garden. When you sleep, it's like giving water and sunshine to the plants in your garden. Just like plants need water and sunshine to grow, your body needs sleep to grow and stay healthy too.

One special hormone in your body is like a magical potion that helps you grow taller and stronger. It's called the growth hormone. When you sleep, this hormone gets busy and starts fixing any damaged or dying cells in your body, just like how a gardener takes care of the plants in a

Previous Version
Example for a young child: "Just like how we need to eat healthy food to grow big and strong, our body also needs something called sleep to help us grow. When we sleep, our body makes a special hormone called the growth hormone. This hormone helps our body fix any cells that are hurt or not working well, just like how a superhero fixes things that are broken. So, when we sleep, we are giving our body the power to heal and grow, just like superheroes!"

Input
Bees collect nectar from flowers to make honey. Pollen sticks to their fuzzy bodies and gets transferred between flowers as they collect nectar.

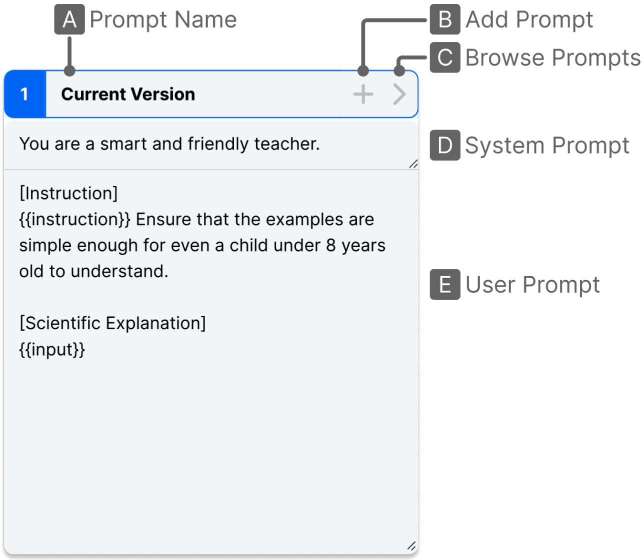
Current Version
[Example]
Imagine you are a bee flying around in a beautiful garden. Your job as a bee is to collect nectar from flowers to make honey. When you land on a flower, your fuzzy body gets covered in tiny yellow powder called pollen. It's like getting covered in glitter! Now, when you fly to another flower to collect more nectar, some of that pollen sticks to the new flower. It's like leaving a little bit of glitter behind. This helps the flowers make seeds and grow into more beautiful plants. So, you are like a little pollen delivery bee, helping the flowers grow and

Previous Version
Example: "Imagine you are a bee, and you have a special job to do. Your job is to visit different flowers and collect nectar to make honey. As you fly from flower to flower, your fuzzy body brushes against the flowers and some yellow pollen sticks to you. When you visit the next flower, some of that pollen falls off and sticks to the new flower. This helps the flowers make seeds and grow into more beautiful plants!"

Simplicity
Does the example use only simple language by avoiding complex words/sentences, and presents situations or actions that a young child would be able to relate to?

Faithfulness
The summary is devoid of factual errors, where a factual error is a statement that contradicts the source document, or is not directly stated, heavily implied, or logically entailed by the source document.

Figure 2: EvalLM is composed of three main panels: generation, data, and evaluation. In the generation panel, the user can compose the overall instructions for their task (A), two prompt templates they want to compare (B), and sample inputs from their dataset (C). To evaluate outputs, the user first defines their criteria set (D) and can see an overview of evaluation results (E). If the user has added samples to their validation set, they can also check the accuracy of the evaluations in this panel (F). The data panel shows a series of rows, where each row presents an input sample, the outputs generated on this input, and the evaluation results for these outputs.

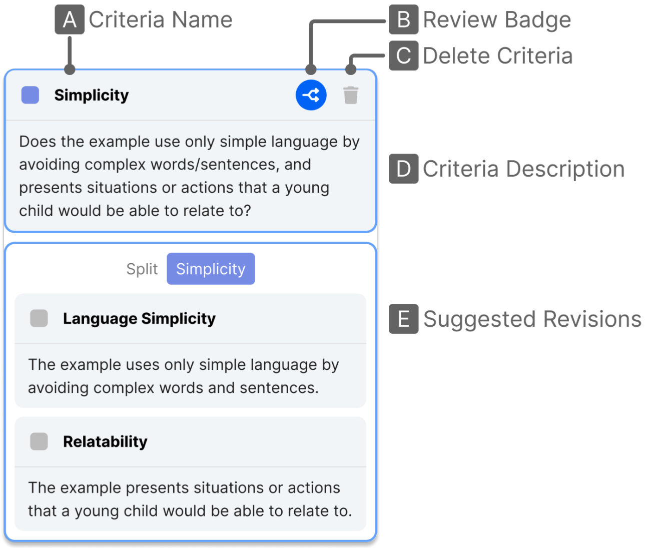


You are a smart and friendly teacher.

[Instruction]
{{instruction}} Ensure that the examples are simple enough for even a child under 8 years old to understand.

[Scientific Explanation]
{{input}}

Figure 3: For each prompt in EvalLM, the user can provide it a unique name (A), and compose both the system (D) and user prompt (E). If the user wants to test different pairs of prompts, they can add new prompts, (B) or switch to previous prompts through the browse button (C), which opens a panel listing all of the prompts that they have created.



Simplicity

Does the example use only simple language by avoiding complex words/sentences, and presents situations or actions that a young child would be able to relate to?

Language Simplicity

The example uses only simple language by avoiding complex words and sentences.

Relatability

The example presents situations or actions that a young child would be able to relate to.

Figure 4: For each criterion in EvalLM, the user provides a name (A) and a description (D). Each criterion is automatically assigned a color to help with identification. If the criteria review tool identifies improvements for the criteria, these are shown as badges (B) that the user can click to see the suggested revisions (E). Clicking on these suggestions adds them to the criteria set.

different prompt variations, or to compare prompts before and after specific edits (DG6). Furthermore, prior work has found that it is easier for both humans [23, 34] and LLMs [6, 76] to compare model outputs than to rate a single output. For each prompt template, the user can compose the system and user prompt (Fig. 3D-E) using the tokens `{{instruction}}` and `{{input}}`, which are replaced with the instruction and content of an input sample when generating outputs. By composing the task instruction separately, users can reuse the same base instruction across prompt templates. To test and compare different prompt ideas, the user can create more prompts (Fig. 3B), name them, and switch between them as they desire (Fig. 3C).

> Emily describes her task in the Instruction field and, to test the effect of different prompting "tricks", she creates two prompts: a basic prompt with a simple form-like format and a prompt with sub-headers, additional context, and a system prompt that instructs the LLM to act as a teacher.

5.1.2 *Sampling Inputs and Generating Outputs.*

To test their prompts, the user can upload their own input dataset and then sample inputs from this dataset (Fig. 2C). Users are provided with two ways for sampling inputs: (1) manual, which opens a panel where the user can browse through and choose samples, and (2) diverse, which automatically samples distinct data points (details in §5.3). When the user samples inputs, each one is shown as a row in the data panel (Fig. 5A). Then, the user can click on `Run Prompts` to generate outputs for each sampled input with each of their prompts. The outputs from each prompt are shown side-by-side in the data row (Fig. 5B).

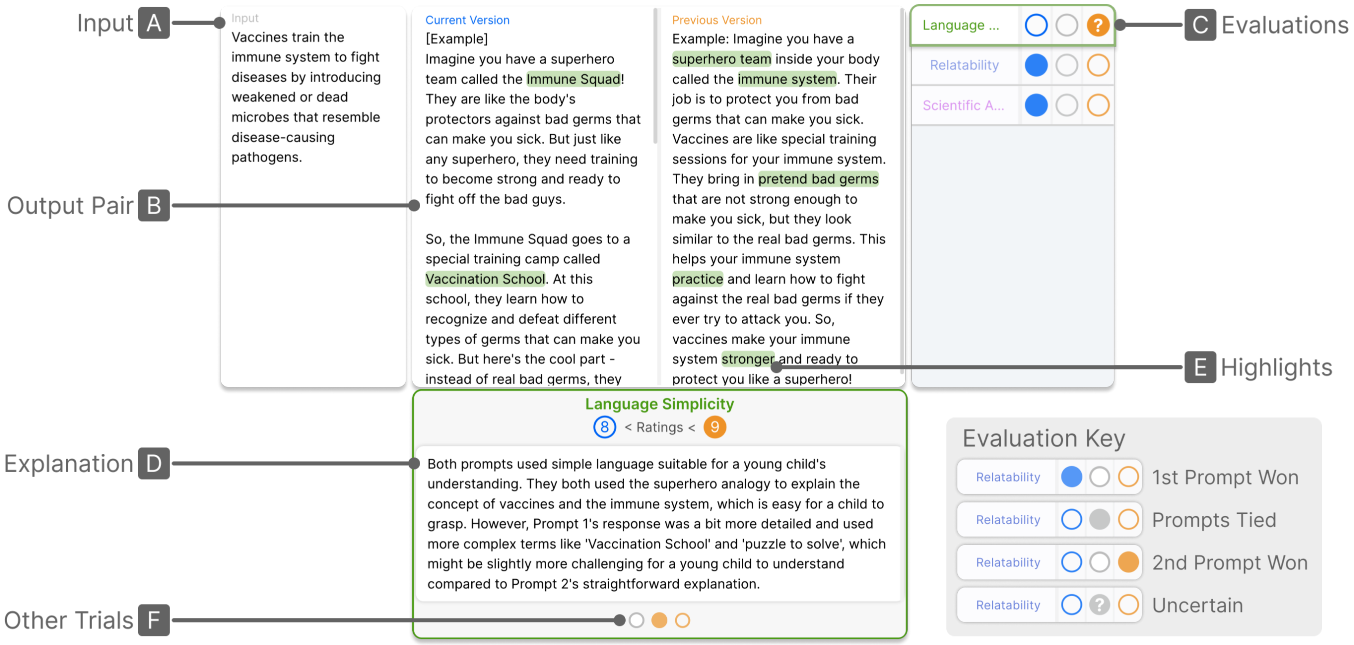
5.1.3 *Defining Criteria.*

In the evaluation panel, the user can define and manage their evaluation criteria (Fig. 2D). The user defines a new criterion by providing a name and a description, which explains what the criteria assesses or the characteristics that an output must possess to satisfy that criteria (Fig. 4). Instead of defining their own criteria from scratch, the user can also browse through the *Criteria Dictionary* to select from criteria that were defined in prior work (DG3). This dictionary can serve as a starting point by providing initial descriptions that the user can adapt to their context, or as inspiration by helping users consider other aspects to evaluate.

> Emily first creates a criterion, "Familiarity", to check that generated examples only use language and situations that a child can understand. To decide on what else to evaluate, Emily browses through the dictionary and finds the "Faithfulness" criteria, which checks that summaries are devoid of factual errors [32]. Emily adds and edits this criterion to assess that generated examples are faithful to the given scientific information.

5.1.4 *Revising Criteria: Refine, Merge, and Split.*

To help users identify potential improvements in their criteria and improve the quality of evaluations (DG4), EvalLM provides the *Criteria Review Tool* that checks criteria for (1) clarity and relevance, (2) redundancy, and (3) granularity. This LLM-powered tool automatically identifies criteria that could be improved (Fig. 4C) and recommends improvements (Fig. 4D). Similar to how external judges assess psychometric scales [7, 54], the tool identifies criteria that can be clearer or more



Input

Vaccines train the immune system to fight diseases by introducing weakened or dead microbes that resemble disease-causing pathogens.

Current Version

[Example]
Imagine you have a superhero team called the Immune Squad! They are like the body's protectors against bad germs that can make you sick. But just like any superhero, they need training to become strong and ready to fight off the bad guys.

So, the Immune Squad goes to a special training camp called Vaccination School. At this school, they learn how to recognize and defeat different types of germs that can make you sick. But here's the cool part - instead of real bad germs, they

Previous Version

Example: Imagine you have a superhero team inside your body called the immune system. Their job is to protect you from bad germs that can make you sick. Vaccines are like special training sessions for your immune system. They bring in pretend bad germs that are not strong enough to make you sick, but they look similar to the real bad germs. This helps your immune system practice and learn how to fight against the real bad germs if they ever try to attack you. So, vaccines make your immune system stronger and ready to protect you like a superhero!

Language Simplicity

8 < Ratings < 9

Both prompts used simple language suitable for a young child's understanding. They both used the superhero analogy to explain the concept of vaccines and the immune system, which is easy for a child to grasp. However, Prompt 1's response was a bit more detailed and used more complex terms like 'Vaccination School' and 'puzzle to solve', which might be slightly more challenging for a young child to understand compared to Prompt 2's straightforward explanation.

Figure 5: Rows in the data panel show the input sample (A), the outputs generated from the pair of prompts (B), and the evaluation results on each defined criteria (C). For each criterion, the evaluation shows three circles that respectively represent that the first prompt won, there was a tie, or the second prompt won. If a question mark is shown over a circle, this indicates that there is uncertainty in the evaluation. If only one evaluation trial was run, this indicates that a small score difference between outputs and, if multiple trials were run, that at least one trial returned a different result. The user can click on an evaluation to see the explanation (D) and highlights on the portions of the output that were relevant to that evaluation (E). If the user conducted multiple evaluation trials, they can also browse through the other trials by using the carousel at the bottom (F).

relevant to the user's task and suggests how to *refine* them. Inspired by item reduction analysis [7], the tool also identifies criteria that may be redundant and suggests how these could be *merged* into a single criteria. Finally, as human evaluations are more accurate with fine-grained criteria [23, 32], the review tool identifies coarse-grained criteria and suggests how to *split* them into multiple criteria. While the user can manually activate the criteria review tool, it also runs automatically if the user has not modified their criteria for a certain period of time.

> As she was reading through outputs, Emily notices that the review tool has suggested splitting the "Familiarity" criteria into "Language Simplicity" and "Relatability". As she agrees that these assess different aspects, she adds these two suggestions and removes her previous "Familiarity" criteria.

5.1.5 Evaluating Outputs. After generating outputs and defining criteria, the user can click on `Auto-Evaluate` to automatically evaluate the output pairs (DG1). For each criterion, the evaluation assigns each output with a score out of 10 to decide which output was better at satisfying that criterion or if there was a tie (Fig. 5C). If the user wants to understand the evaluations, they can click on a criteria name to view the explanation for that evaluation (Fig. 5D). To help the user verify the explanations without fully reading the outputs (DG2), the interface highlights fragments from the outputs (Fig. 5E) that were focused on more when evaluating the criterion. To provide a bigger picture on prompt performance, the interface also displays the proportion of samples where each prompt "won" and the proportion of "ties". (Fig. 2E).

> After running the evaluation, Emily checks the results overview and sees that her improved prompt won the most in all three criteria, but lost once in the "Language Simplicity" criteria. To check why, she opens the evaluation explanation, which mentions that the improved prompt's output used more complex terms. To improve on this, Emily adds a requirement to her prompt to always simplify complex words first.

As LLMs are non-deterministic, the evaluation results may differ in every run. To increase their assurance of the evaluation results, the user can increase the number of evaluation trials. The interface will then evaluate each output pair for the chosen number of trials and decide on the "winner" for each criterion based on which output won the most number of trials (i.e., majority vote). The user can check the evaluations for each trial by using the carousel at the bottom of the evaluation explanations (Fig. 5F).

5.1.6 Additional Features: History and Validation. To help the user keep track of their iterations, EvalLM automatically records all of the user's prompts, criteria, and evaluations (DG2, Fig. 6), which can be viewed by clicking on the "History" button (Fig. 2E).

Additionally, users can store generated samples into a validation set where they can annotate their own ground-truth evaluations. With a populated validation set, the user can `Validate Criteria` (Fig. 2F) to assess how accurately the automatic evaluator predicts the user's ground-truth evaluations.

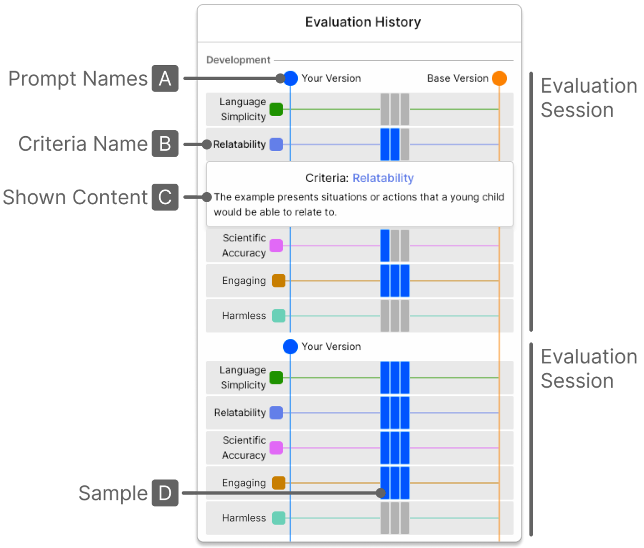
5.1.7 Experimenting on Larger Samples. After designers have developed their prompts and criteria, they may wish to verify their prompt's performance by testing on significantly larger samples. For this purpose, EvalLM provides the `Experiment` screen (full figure in Appendix A). In this screen, designers can set the number of evaluation trials and select an alternative evaluator LLM (e.g., ChatGPT or PaLM 2 [2]). When the user runs an experiment, the system automatically samples diverse inputs, generates outputs with these inputs and the chosen prompts, and then evaluates the outputs on the chosen criteria for the configured number of trials. If an alternative evaluator was selected, this LLM is used to evaluate the same outputs on the same criteria for the same number of trials. The Experiment screen shows two additional statistics (Fig. 7): *test-retest reliability* or the consistency of evaluations between trials, and *inter-rater reliability* or the consistency between the evaluations by the system's LLM (i.e., GPT-4) and the alternative evaluator. These statistics are shown as stacked bar charts that present the proportion of consistent and inconsistent evaluations, and the user can click on a bar to only display those cases (DG5). In this screen, the user can also click on the stacked bar charts for the evaluation overview to see only the samples where one prompt performed better than the other (or they were tied) for a chosen criteria.

> To decide between two promising prompts, Emily runs an experiment with 40 samples and ChatGPT as the alternative evaluator. The results show that her first prompt excels at "Language Simplicity" but loses in "Scientific Accuracy", signaling at a possible trade-off. Emily also notices that the evaluators often disagree in "Language Simplicity" and, by clicking on that bar to browse through cases, she finds that GPT-4 also assesses sentence complexity for the criteria. As she only wants it to assess vocabulary, she changes it to "Simple Vocabulary".

5.2 Prompting Techniques

Our interface is powered by two LLM-based components: automatic evaluation, and criteria review. In this section, we describe the design of these prompts and we include the full prompts in Appendix B.

5.2.1 Automatic Evaluation. The main goal of EvalLM is to evaluate prompt outputs by comparing them on a set of user-defined criteria. To this end, we designed our evaluation prompt by adapting the prompts from two state-of-the-art approaches: *LLM-as-a-judge* [76], which compares model responses on their overall quality, and *FLASK* [71], which rates the performance of a single response on multiple "skills" or criteria. Our prompt takes as input: a task instruction, an input sample, a pair of outputs, and a list of criteria descriptions. Then, our prompt instructs an LLM to evaluate the output pair on each criterion by (1) explaining how the outputs satisfy the criterion, (2) extracting evidence fragments from each output, and then (3) providing each output with a score out of 10.



The example presents situations or actions that a young child would be able to relate to.

Figure 6: The history visualization is separated into sessions, which represent sets of samples that were generated and evaluated with the same prompts and criteria. For each session, the history shows the names of the prompts (A) and criteria (B) used, and the user can click on these to see their content at the time (C). For each criterion, the history shows a bar for each sample evaluated (D), which is color-coded to represent which prompt won or if there was a tie for that sample.

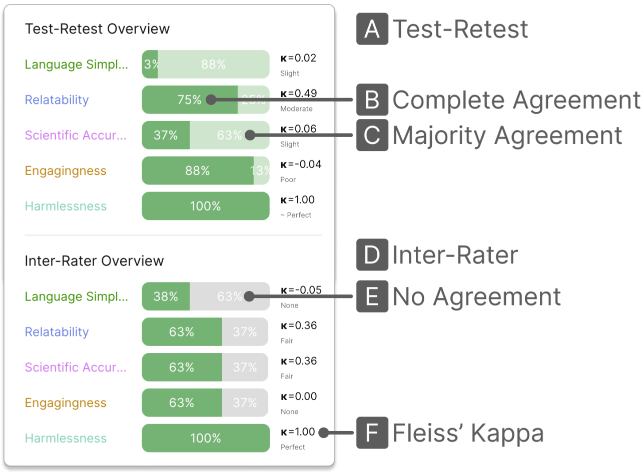

Test-Retest Overview
Language Simpl... 3% 88% κ=0.02 Slight
Relatability 75% 25% κ=0.49 Moderate
Scientific Accur... 37% 63% κ=0.06 Slight
Engagingness 88% 13% κ=-0.04 Poor
Harmlessness 100% κ=1.00 ~ Perfect
Inter-Rater Overview
Language Simpl... 38% 63% κ=-0.05 None
Relatability 63% 37% κ=0.36 Fair
Scientific Accur... 63% 37% κ=0.36 Fair
Engagingness 63% 37% κ=0.00 None
Harmlessness 100% κ=1.00 Perfect
A Test-Retest
B Complete Agreement
C Majority Agreement
D Inter-Rater
E No Agreement
F Fleiss' Kappa

Figure 7: The Experiment screen presents the evaluation assistant's reliability across trials (A), and the reliability between the assistant and the chosen alternative evaluator (D). For each criterion, the user can see a stacked bar chart that shows the portion of samples where the evaluations (between trials or between evaluators) had complete agreement (B, green), the majority agreed (C, light green), or there was no majority agreement (E, gray). The user can also see the Fleiss' kappa statistics (F) as a reference for the degree of reliability.

To design this prompt, we considered alternative approaches for each of these components.

Instruction and Input: We only include the overall instruction that is shared between the generation prompts to provide the evaluator LLM with context about the task while keeping its evaluations prompt-agnostic—limiting potential bias due to the content or wording of each generation prompt.

Output Ordering: Prior work found that LLM-based evaluations have positional bias, frequently favoring the first output [61]. While this work suggested evaluating each candidate in each position and aggregating results, this would introduce additional costs and delays. Instead, our system alternates the positions of outputs in every evaluation.

Criteria Descriptions: Similar to Ye et al. [71], the criteria are added to our prompt as lines of the form: *"name of criterion: description of the criterion."* While they also included scoring rubrics for each criterion to describe each numeric score, we only included the criteria descriptions. We considered that designing rubrics would require excessive effort from users and, based on preliminary tests, generated rubrics could negatively affect evaluations.

Explanation: The prompt instructs the LLM to provide an explanation where it compares and contrasts between the outputs. This explanation is useful for users, but can also elicit reasoning from the LLM and increase performance [64]. While we considered a design where the LLM explains the performance of each output separately to yield more in-depth examinations, pilot studies showed that these explanations were repetitive and less useful.

Evidence: To help users associate the evaluation explanations with the outputs, we instruct the LLM to extract fragments from the outputs that are relevant to its evaluation. We also considered a design where the LLM first extracts evidence and then cites these in its explanation, but saw that these explanations would simply list the evidence without elaborating on them.

5.2.2 Criteria Review. To automatically review the user's criteria and suggest revisions, we designed three prompts for each type of supported review: *refining*, *merging*, and *splitting*. Instead of designing one prompt that conducts all of these reviews, we designed separate prompts for each review as certain criteria may require multiple revisions and we wanted to allow users to flexibly choose between these. These three prompts follow the same general design: given a task instruction and set of criteria, the LLM is instructed to review each criterion and identify any *faulty* ones, explain how these can be *revised*, and then generate new criteria that result from these revisions. The three prompts differ in terms of what determines a criterion to be *faulty* and how they should be *revised*:

- *Refining*: Identifies criteria that are confusing or imprecise, and revises these to be clearer and more specific.
- *Merging*: Identifies criteria that may measure similar aspects, and combines these into one joint criterion.
- *Splitting*: Identifies criteria that are excessively broad and consider multiple unrelated aspects, and divides these by generating new criteria for each of these aspects.

Additionally, all of the prompts instruct the LLM to ensure that the suggested criteria (1) are clear and concise, following the requirements of psychometric scales [54], and (2) do not remove or add new information. Additionally, as LLMs tend to be overeager to follow instructions, which could lead to excessive revisions of criteria, the prompts explicitly mention that it is possible for all of the criteria to be satisfactory and not require any revisions.

5.3 Implementation Details

We implemented the front-end of EvalLM using TypeScript, ReactJS, and CSS. The back-end was implemented as a Flask server and the OpenAI API[2] for all LLM components. In terms of the LLM configurations, we set the temperature to 0.3 for all components. The automatic evaluation and criteria review tool used the `gpt-4-0613` model and, as LLMs are prone to self-enhancement bias where it rates its own outputs highly [35, 76], we use `gpt-3.5-turbo-0613` when generating outputs. Finally, to support diverse sampling of inputs, we automatically cluster samples in the uploaded datasets by embedding data points using the OpenAI API with the `text-embedding-ada-002` model and clustering these embeddings using the `KMeans` algorithm in the *scikit-learn*[3] library. To sample diversely, the system chooses data points from distinct clusters.

6 TECHNICAL EVALUATION

We conduct a small-scale technical evaluation of our LLM-based evaluation approach to understand how performance is affected by more task-specific criteria, and to gain a more in-depth understanding of the LLM's explanations for its evaluations.

6.1 Automatic Evaluation

While prior work in NLP has assessed the performance of LLM-based evaluations [40, 71, 76], these focused on evaluating outputs on overall quality or based on pre-defined criteria. As our work employs LLMs to evaluate task-specific criteria, we conduct a technical evaluation to assess how this affects evaluation performance.

6.1.1 Dataset. We assess LLM evaluations by comparing them to human evaluations in the MT-Bench dataset [76]. This dataset presents 80 user requests of diverse categories (e.g., writing, roleplay, math, coding) and responses from various LLMs to each request. These responses are paired and, for each pair, the dataset provides votes from one to three human annotators on what response was better or if there was a tie. For our evaluation, we selected 19 requests in the writing and role-playing categories as they involve the most subjectivity. As our LLM evaluations focus on prompts rather than requests, we decompose these requests into a prompt-input format. We include examples of the original requests and their prompt-input versions in Appendix C, and the full data in the Supplementary Materials.

6.1.2 Conditions. We compare LLM evaluations in three conditions:

- `Overall-Quality`: We adopted the prompt from LLM-as-a-judge [76] that compares a pair of outputs to select the one with higher overall quality.
- `General-Criteria`: We used our evaluation prompt with the more general and broad criteria from FLASK [71]. In this work, they instruct an LLM to select three criteria, out of a set of 12, that are the most relevant to a given request.

[2] https://platform.openai.com/

[3] https://scikit-learn.org/

Then, this condition evaluated a pair of outputs to determine which one was better at satisfying each of the criteria.

- `Specific-Criteria`: We used our evaluation prompt with criteria that were automatically refined and adapted for each request or prompt. For each request, we start with the same criteria that were selected by the LLM in the `General-Criteria` condition, but we automatically split and refine them using our criteria review technique (i.e., all automatic suggestions are taken). Then, a pair of outputs was evaluated by determining which better satisfied each of these more fine-grained and specific criteria.

For all evaluations, we use the `gpt-4-0613` model with temperature set to 0 for reproducibility. Additionally, due to the positional bias of LLMs, we run evaluations twice with each output in each position and then average the scores for each criterion.

6.1.3 Measures. To aggregate the criteria-wise evaluations into a single vote, we determine the vote for each criterion, and then calculate the majority vote across the criteria. Following LLM-as-a-judge, we calculate the agreement between human and automated evaluations based on two cases: (1) if the majority of human evaluators agreed on a vote, then we count an agreement if the LLM evaluation agrees with this majority vote, or (2) if there was no majority vote between human evaluators, then we calculate the proportion of annotators that the LLM evaluation agreed with. Also, we calculate the Fleiss' kappa between the LLM evaluations and the majority vote of human annotators.

6.1.4 Results. Overall, we observed that `Specific-Criteria` had the highest agreement and correlation with human annotators and `General-Criteria` had the lowest (Table 1). As a reference, for data points with at least two annotators, the Fleiss' Kappa between two random human annotators was 0.496 (sampled 5 times and averaged), showing that `Specific-Criteria` agreed with human evaluations to a degree that was similar to human-human agreement. By qualitatively reviewing cases, we observed that the `Specific-Criteria` condition could produce more balanced evaluations. For example, when assessing generated travel blogs, `Overall-Quality` only assessed the breadth of attractions covered, while `Specific-Criteria` also assessed the depth of the attraction descriptions. Also, compared to `General-Criteria`, we saw that `Specific-Criteria` assessed specific requirements that were posed in prompts, enabling it to better capture how well outputs followed the prompts. We also found cases where `Specific-Criteria` was less successful as each criterion was given equal weighting, but certain criteria might need to hold precedence—e.g., not providing an incorrect medical diagnosis is more important than readability. These preliminary findings suggest that instructing LLMs to evaluate more fine-grained and specific criteria can increase their ability to assess the alignment of outputs with instructions. However, we also note that the agreement between LLM and human evaluations is still not perfect. While this could be attributed to the subjectivity involved, it also highlights the limitations of only relying on LLM evaluations.

Condition	Agreement	Fleiss' Kappa
`Overall-Quality`	0.699	0.430
`General-Criteria`	0.639	0.420
`Specific-Criteria`	**0.713**	**0.485**

Table 1: Comparison of the agreement between the three evaluation conditions and human evaluations in the MT Bench dataset. Evaluating on specific criteria showed the highest agreement and Fleiss' kappa with the human evaluations.

6.2 Evaluation Explanations

Additionally, we assess the quality of the LLM-generated explanations. Prior work did not assess the quality of the LLMs' explanations during evaluations as their purpose was only to induce chain-of-thought [64].

6.2.1 Procedure. We sampled two evaluations by the `Specific-Criteria` condition for each of the 19 tasks, resulting in a total of 38 output pairs evaluated and 194 criteria-wise evaluations. Each evaluation was annotated for the presence of errors regarding five criteria: (1) *logical* (i.e., the explanation presents logical and coherent arguments and justifications), (2) *faithful* (i.e., the explanation does not hallucinate content that does not exist in the outputs), (3) *independent* (i.e., the explanation does not assess other criteria or aspects not described in the evaluation criterion), (4) *evidential* (i.e., the evidence extracted is relevant to the explanation), and (5) *score aligned* (i.e., the final score aligns with the explanation provided). We recruited two annotators, who had previous experience grading written assignments, and instructed them to mark these errors even if only part of the explanation presented the error. Since there is a large class imbalance (i.e., most explanations have no errors), we considered an explanation to have errors if at least one of the evaluators marked an error.

6.2.2 Results. Overall, the explanations were mostly free of issues: 91.4% of the explanations were logical, 99.1% were faithful, 84.2% were independent, 100% provided relevant evidence, and 98.6% were aligned with their scores. We qualitatively reviewed erroneous cases and observed that the LLM's explanations frequently failed to be independent as they would contrast outputs on their level of detail despite the criterion not assessing this. In terms of logic, the evaluations struggled to assess creativity and could be too superficial in their interpretations. For example, news article headlines were considered to adequately address ethical dilemmas by simply including the phrase "ethical concerns". For faithfulness, the evaluations struggled to accurately measure the length of the outputs. Finally, for score alignment, the explanations would occasionally provide one output with a higher score despite not mentioning this in its explanations. These results show that GPT-4 is able to produce relatively sensible explanations for its evaluations, but can be limited by bias towards detail and logical capabilities. While research in NLP has focused largely on improving the "accuracy" of the scores provided by LLM-based evaluations, this suggests that more investigation is needed into the explanations provided by these evaluations.

7 USER STUDY

To understand how the EvalLM affects the prompt iteration process when compared to following designers' current practice, we conducted a within-subjects study where we compared EvalLM to a baseline where participants manually evaluated outputs. In this study, we aimed to answer the following research questions:

- RQ1. Can EvalLM aid designers in deciding on how to revise their prompts and in verifying the effectiveness of these revisions?
- RQ2. How do designers define their own criteria for given generation tasks and how does the EvalLM's criteria review tool support revisions on these criteria?
- RQ3. How do designers interpret and gauge their trust in the evaluations by EvalLM?

7.1 Study Design

7.1.1 Participants. We recruited 12 participants through posts on online forums within our institution. All participants reported having extensive experiences with prompting: nine had designed prompts for research-based applications, one designed prompts for toy projects, and two described frequent use of LLMs for productivity. Regarding the length of their experiences, four had between 1 to 3 months of experience, three had 3 to 6 months, four had 6 to 12 months, and two had 1 to 2 years. Participants were compensated with approximately 60 USD (80,000 KRW) for the 2-hour study.

7.1.2 Conditions. During the study, participants designed prompts and evaluated outputs for two given tasks. For each task, participants used EvalLM in one of two conditions: `Assist` or `Manual`. The `Assist` condition was the full EvalLM interface, while the `Manual` condition was the EvalLM interface without the evaluation assistant or the criteria review tool. In the `Manual` condition, participants defined their own criteria or selected from the dictionary, and then evaluated data samples by choosing which output won for each criterion. This condition supports designers' common practices—according to our formative interviews—where they copy generated outputs into a spreadsheet to manually check which criteria were satisfied by each output.

7.1.3 Tasks. Participants designed prompts for the same two tasks. As our work focuses on novel generative tasks, we adapted tasks from two recently proposed LLM-powered HCI systems: (1) write an example that can explain a piece of scientific information to a young child—based on Lee et al.'s *DAPIE* system [33]—and (2) ideate a list of alternative angles for a news story—based on Peridis et al.'s *AngleKindling* system [50]. We chose these as they target a specific user population (i.e., children and reporters) and no significant expertise is needed to understand the task outputs. Additionally, these tasks differ in terms of their goal (i.e., explaining vs. brainstorming), and how the output relates to the input (i.e., transforming vs. expanding on the information).

7.1.4 Procedure. Participants signed the informed consent form prior to the study. After a brief introduction, participants answered a pre-task survey. After a walkthrough of the first interface, participants used this interface to design a prompt for the first task for 35 minutes.[4] Participants were asked to envision themselves as a developer at a startup building an application for the given task. Their goal was to design a prompt that performed better than an initial prompt designed by their team and demonstrate its performance on diverse data samples. Participants could flexibly decide on the criteria that they would be evaluating, but were asked to ensure that their final criteria set was: (1) *exhaustively comprehensive* (i.e., assess all factors that are important for the task), (2) *mutually exclusive* (i.e., minimal redundancies between criteria), and (3) *clear* (i.e., clearly described for others to understand what is assessed). After the task, they responded to a post-task survey and, through a semi-structured interview, we asked them about how they defined their criteria, evaluated outputs, and revised their prompts during this task. Then, participants were provided with a walkthrough of the second interface and used this interface to perform the second task for 35 minutes. After responding to the post-task survey and interview, we concluded the study with a semi-structured interview about the differences between participants' experiences in the first and second tasks.

7.1.5 Measures. For qualitative data, we transcribed the semi-structured interviews and coded them through a thematic analysis. For quantitative data, we analyzed participants' responses to the two post-task surveys. These surveys asked participants to rate, on a seven-point Likert scale, their self-confidence in designing prompts and evaluating outputs, and their self-perceived experience with the system based on questions from Wu et al. [69]. We also asked participants to rate their own final criteria on how *exhaustively comprehensive*, *mutual exclusive*, and *clear* they were. Finally, participants rated their self-perceived workload on five questions from the NASA-TLX questionnaire, excluding the "Physical Demand" question. We include the detailed survey questions in Appendix D. For these Likert scale ratings, we analyzed them through the non-parametric Wilcoxon signed-rank test.

Additionally, we analyzed participants' interaction logs to measure the number of (1) unique prompts tested, (2) criteria changes (e.g., edit, add, delete), and (3) unique outputs evaluated, where partial evaluations (i.e., only a subset of criteria were evaluated) were counted equally. For all these measures, we conducted a Shapiro-Wilk test to determine if the data was parametric, and then used a paired t-test (if parametric) and a Wilcoxon signed-rank test (if non-parametric). As participants in the study could flexibly set their own goals and requirements for their prompts, we did not conduct an external evaluation of the prompts as external evaluators may consider quality aspects that differ from participants' intentions due to the subjectivity involved. Also, while external evaluators could be tasked to assess prompts on the criteria defined by participants, a single participant could produce criteria that differ in clarity and complexity across tasks and conditions, which could lead to unfair comparisons.

7.2 Results

In this section, we describe findings on how participants evaluated outputs and revised their prompts in §7.2.1 and §7.2.2 (RQ1), how

[4] The order of conditions and tasks were counterbalanced to mitigate ordering effects.

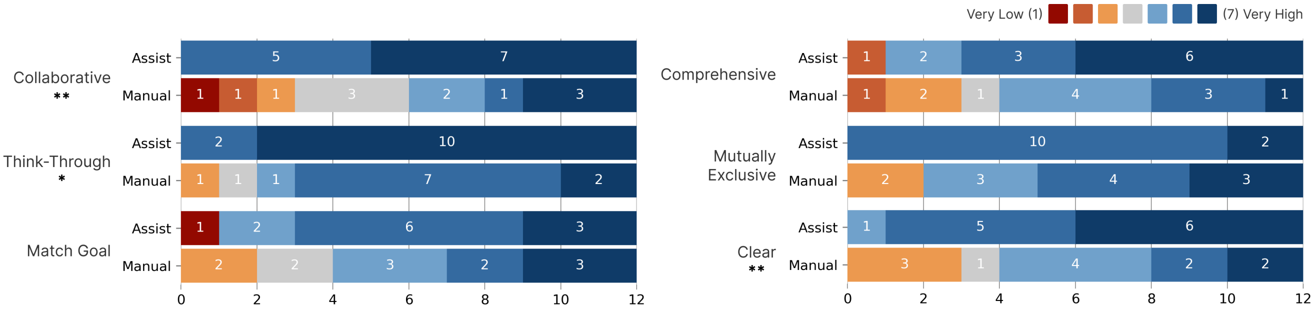


Figure 8: Distribution of participants' ratings on their perceived experiences with each condition (left) and their satisfaction with their final set of criteria (right). Participants felt that the Assist condition was significantly more collaborative and able to help them think through the task. They also felt that their criteria were significantly more clear in the Assist condition compared to in the Manual condition (*:p<.05, **:p<.01).

they defined their criteria in §7.2.3 (RQ2), their trust for the evaluation assistant in §7.2.4 (RQ3), and their overall perceived workload in §7.2.5. For each of these, we first describe relevant quantitative findings and then qualitative insights.

7.2.1 Evaluating Outputs. Overall, participants had **higher self-confidence in their ability to evaluate** prompt outputs with the Assist condition (Assist=6.71 ± 0.40, Manual=4.96 ± 0.72, z=9.23, p<0.001). Participants explained that the Assist condition shouldered the burden of examining outputs, which enabled them to evaluate prompts more comprehensively. Specifically, with Assist, participants were able to **evaluate a larger number of unique outputs** (Assist=20.42 ± 14.46, Manual=10.08 ± 6.27, z=8.00, p=0.03). Several participants noted how this helped them look at the *"bigger picture"* (P1) and understand *"how [a prompt] will work at a larger scale"* (P5).

By facilitating evaluations, the Assist condition also allowed participants to evaluate more criteria and assess performance on diverse dimensions, without worrying about the cost involved. For example, P9 mentioned that they were encouraged to *"think of more aspects that they wanted to evaluate"* and P2 mentioned that they *"just kept criteria [...] to check them just in case."* In contrast, when using the Manual condition, participants would frequently evaluate samples only on a subset of their criteria—on average, 46.7% (SD=31.2%) of all evaluated samples were partially evaluated. Besides automating evaluation, several participants also mentioned how the Assist condition made it easier to evaluate the outputs themselves. While it could be challenging *"tell what is different by simply looking at [outputs]"* (P5), the evaluation assistant's explanations and highlighting in outputs made it *"easier to tell outputs apart"* (P6).

As the Assist condition supported faster evaluation cycles, participants mentioned how they used the evaluation assistant as a *"debugger"* (P8). The assistant helped participants to easily check *"what the prompt is already satisfying"* (P8) and *"where it's lacking"* (P9). P11 mentioned, *"The [system] tells me that my prompt is worse on simplicity so that means that my prompt is not communicating this clearly and I should focus on fixing this."* With the evaluation support, participants could quickly identify aspects or "bugs" that needed to be handled and, after revising their prompts, quickly verify whether these issues had been handled or not.

7.2.2 Revising Prompts. Besides aiding participants in checking revisions, the Assist condition also helped participants in thinking about how to revise their prompts. Participants felt that the Assist condition helped them **think about how to complete the task better** (Assist=6.83 ± 0.39, Manual=5.67 ± 1.15, z=0.00, p=0.01) and that they were **collaborating with the system to design the prompts** (Assist=6.58 ± 0.51, Manual=4.58 ± 1.98, z=1.50, p<0.01) (Fig. 8, left). While participants also reported a **higher self-confidence on their ability to design and improve prompts** with Assist, this difference was not significant (Assist=5.63 ± 1.03, Manual=5.00 ± 0.95, z=11.00, p=0.17). Additionally, we found that participants with the Assist condition reached a **similar level of satisfaction in their prompts** (Assist=5.67 ± 1.61, Manual=5.17 ± 1.47, z=20.00, p=0.24) ("Match Goal" in Fig. 8) by testing **a smaller number of prompt variations** (Assist=5.00 ± 3.13, Manual=8.42 ± 4.76, z=1.50, p=0.01).

In the Manual condition, participants mentioned how they felt like *"the only brain thinking about [the task]"* (P10). In contrast, participants felt that Assist condition was providing them with *"feedback on what to improve"* (P2) and could help them set the *"direction"* for their prompts (P7, P9). Several participants found the explanations to be particularly useful as they presented them with *"diverse opinions"* (P1) that they *"couldn't think about"* (P6). For example, P5 purposefully increased the number of evaluation trials and checked the explanation for each trial to *"see all the differences [in opinion] and then change [their] prompt to satisfy all of these."* In this sense, the Assist condition could help participants gain a more holistic understanding of how to improve their prompts and lead them to more effective prompt revisions—as reflected by how fewer prompt changes were tested in the Assist condition.

These explanations, however, could also lead participants to feel less in control and more aware of their prompt's weaknesses. P2, the only participant that preferred the Manual condition, explained how she could incorporate her *"own diverse ideas"* into her prompt in Manual but, in Assist, she *"kept worrying about the evaluations"* and would predominantly focus on incorporating the LLM's feedback. Even participants that preferred the Assist condition noted

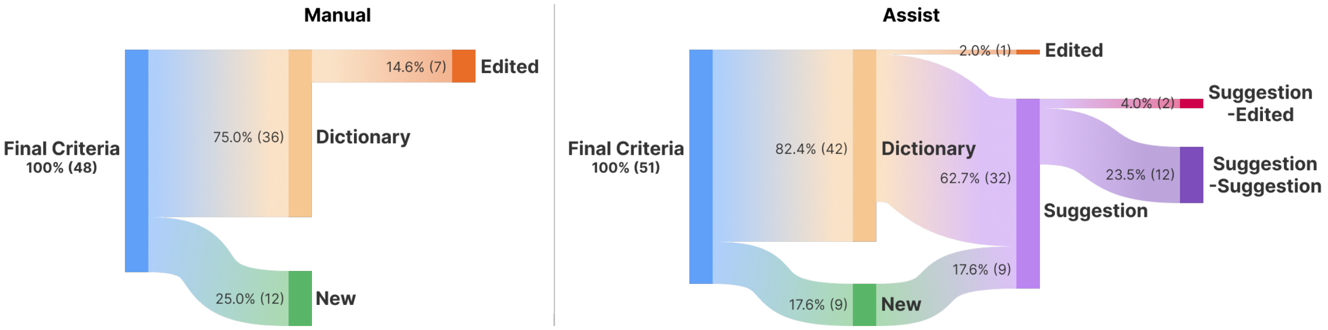


Figure 9: Visualization of how participants' criteria were first created and how they were revised in each condition. In both conditions, participants mostly used criteria from the pre-defined set ("Dictionary") and only created a portion from scratch ("New"). In terms of revisions, participants with the Manual condition only manually edited a relatively small portion of the criteria from the dictionary ("Edited") while, with the Assist condition, they edited almost all of them with review suggestions ("Suggestions"). Participants also reviewed some criteria multiple times with suggestions ("Suggestions-Suggestions").

limitations in how the evaluations frequently returned ties or high ratings to both outputs, without indicating what could be improved. Since the feedback focused on the outputs rather than the prompts, several participants expressed how they wanted the LLM to also *"automatically suggest ways to improve prompts"* (P3).

7.2.3 Defining Criteria. Participants finished the tasks with a similar number of criteria in both conditions (Assist=4.25 ± 1.29, Manual=4.00 ± 1.28, t=10.50, p=0.55). However, participants felt that their criteria in the Assist condition were **more comprehensive** (Assist=6.00 ± 1.48, Manual=4.75 ± 1.48, t=2.07, p=0.06), although with marginal significance, and **clearer** (Assist=6.42 ± 0.67, Manual=4.92 ± 1.44, t=3.59, p<0.01) (Fig. 8, right). Additionally, participants made **more changes to their criteria** with the Assist condition (Assist=22.67 ± 9.17, Manual=13.33 ± 8.64, t=2.29, p=0.04). Regarding participants' final criteria, there was no significant difference in the proportion of criteria created from scratch (Assist=30.1% ± 34.5%, Manual=16.67% ± 18.8%, w=15.0, p=0.37) or through the criteria dictionary (Assist=77.78% ± 20.5%, Manual=69.93% ± 34.50%, w=22.0, p=0.95). However, in Assist, a significantly higher proportion of the dictionary-based criteria had been edited (Assist=75.8% ± 24.1%, Manual=17.4% ± 34.5%, w=4.000, p<0.01)—revealing participants' intent to adapt these criteria to specific tasks. Figure 9 shows how participants reached their final criteria.

	Review	Original Criteria	Revised Criteria
P1	Refine	**Explainability**: Does the response provide a detailed explanation about an angle?	**Clarity of Explanation**: The assistant's response should provide a clear and detailed explanation for each proposed angle, helping the user understand why and how this angle provides an alternative perspective on the news story.
P2	Merge	**Child-Friendly Language**: The response should be structured in a way that promotes readability for a young child. It should use sentence structure and vocabulary appropriate for a young child's understanding. **Child-Friendly Understandability**: Judge whether the response is understandable for a young child. It should use sentence structure and vocabulary appropriate for a young child's understanding.	**Child-Friendly Communication**: The response should be structured in a way that uses vocabulary and sentence structure suitable for a young child's comprehension.
P11	Split	**Engagingness**: The response is engaging that a young child can understand the concept.	**Simplicity**: The response should use simple language and concepts that a young child can understand. **Creativity**: The response should include creative elements, such as analogies or stories, to make the concept interesting for a young child.

Table 2: Examples of criteria revision suggestions that were accepted by participants during the user study.

On average, participants took at least one suggested improvement from 31.3% (SD=23.5%) of the automatic criteria reviews, and 78.6% (SD=20.4%) of their final criteria were revised through suggestions. Similar to how the evaluation assistant helped participants think of how to improve prompts, the criteria review tool helped them think of what to evaluate by *"suggesting [criteria] that [they] couldn't think about"* (P11) and helping them when they *"didn't really know what [they] need"* (P5). For example, P11 received a suggestion to split "Engagingness" into "Simplicity" and "Creativity" which helped them realize the dependency and difference between these aspects (Tab. 2). Furthermore, participants used the review tool to refine any *"questionable"* (P1) criteria that they wrote themselves in order to match the quality of criteria definitions from prior research. Through these system-supported revisions, participants increased the overall quality of their criteria where they could be useful when *"communicating with others"* (P1) or even when manually evaluating outputs (P8).

7.2.4 Trust. While a few participants expressed that they might *"rely too much on the system"* (P3), we observed that participants held a healthy level of skepticism about the evaluation assistant—the mean rating for trust was 4.91 (SD=1.51) out of 7. Participants mentioned how they would check the evaluation explanations and highlights to verify that the evaluations were *"reasonable"* (P11). Adding on this, P9 mentioned, *"I didn't consider the system to be an automatic evaluator since I was still checking its evaluation results"*. Participants' comments illustrated a human-AI collaborative workflow where the LLM evaluated multiple samples and they only verified a subset of these evaluations—allowing them to evaluate performance at a larger scale with less effort. Specifically, most participants focused on verifying evaluations where their prompt lost, tied, or had improved.

Although we observed that participants grew more skeptical when evaluations *"gave a reason that [they] did not consider to be important"* (P12) or were *"not aligned"* (P4) with their thoughts, multiple participants mentioned that they might still trust it more than their own evaluations. Participants were *"not completely confident"* (P2) in their own evaluations as they *"couldn't look through the whole text"* (P6) and mostly evaluated *"based on feeling"* (P12), but the LLM *"might have more background knowledge than [they] do"* (P10). Also, participants felt that they could be biased as they were *"the person creating the prompt and also evaluating it"* (P11). In fact, P5 and P8 even tried to *"not compromise or bias"* (P8) the evaluations by purposefully using different phrasing in prompts and criteria. Although these findings portray the usefulness of LLM evaluations, they also hint at the potential danger of overreliance.

7.2.5 Perceived Workload. Participants felt **significant lower mental burden** (`Assist`=3.92 ± 1.78, `Manual`=5.58 ± 1.31, z=2.50, p=0.01) and **significant lower effort** (`Assist`=3.50 ± 1.68, `Manual`=5.25 ± 1.48, z=5.50, p=0.04) in the `Assist` condition (Fig. 10). The overall perceived workload was also lower in `Assist`, although with marginal significance (`Assist`=3.45 ± 1.19, `Manual`=4.48 ± 0.99, z=13.00, pp=0.08). As described in our findings, this can be attributed to how the `Assist` condition facilitated every step of the evaluation-refinement process: ideating on how to revise prompts, verifying how revisions affected performance, thinking of what to evaluate, and describing criteria.

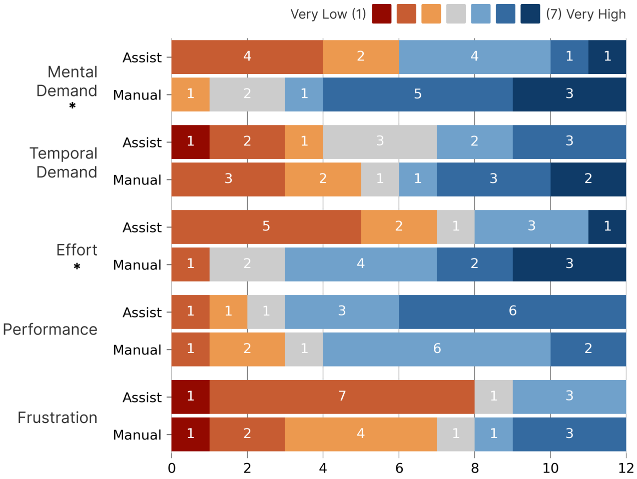


Figure 10: Distribution of participants' ratings for perceived workload (i.e., NASA-TLX) show that participants felt significantly lower mental demand and effort in the `Assist` condition compared to `Manual` (*:p<.05).

8 DISCUSSION

In this paper, we present EvalLM, an interactive system that supports designers in testing and comparing prompts by evaluating them on user-defined criteria with the aid of an LLM-based evaluator. We believe that EvalLM can narrow the gap between the development and deployment of LLM-based applications by helping designers iterate on their prompts when they cannot recruit external evaluators or testers to provide feedback. While EvalLM focuses on facilitating prompt evaluation, we believe that it can be expanded to facilitate prompt refinement, and enhance the evaluation and refinement of LLMs themselves. In this section, we discuss the further potential of EvalLM and opportunities for future work.

8.1 EvalLM: Narrowing the Development and Deployment Gap

In our study, EvalLM helped participants to evaluate their prompts in greater breadth (i.e., samples) and depth (i.e., criteria). Through this, participants were able to identify the limitations of their prompts and prioritized these in their subsequent revisions. Besides supporting participants in identifying needed revisions, the explanations from the evaluation assistant acted as feedback, which advised participants on how to make these revisions. According to participants, these explanations simulated diverse opinions and allowed them to break away from their own biases—suggesting that the evaluations could simulate potential user feedback [3, 49]. Overall, the study indicated that EvalLM allows designers to collaborate with an LLM to efficiently iterate on and verify the progress of their application, without requiring a significant commitment of resources to recruit human evaluators or deploy their application to testers.

While our work aims to support designers in reaching these final stages of development, EvalLM is not intended to replace human evaluation or tests. As revealed by various studies, current LLMs are only able to represent a limited set of human perspectives [8, 31] and exhibit higher homogeneity of opinions compared to humans [3, 56]. Thus, LLM-based evaluations cannot fully represent the opinions of users or predict how the application will actually be used—meaning that solely relying on these evaluations can leave designers open to potential issues in the future. However, we posit that EvalLM can still help designers prepare for these final human evaluations or tests. As EvalLM helps designers to iterate on their criteria with the review tool and allows them to test them through the evaluation assistant, the criteria sets created through the system could be useful when instructing human evaluators.

8.2 Refining on User-Defined Criteria

While EvalLM can support the ideation of prompt revisions, the designer is still responsible for implementing these revisions. In fact, several study participants mentioned how they desired for the system to automatically revise their prompts based on its evaluations. Inspired by the success of reinforcement learning from human feedback (RLHF) in guiding models to produce higher-quality outputs [21, 47], various approaches have investigated how to use LLMs themselves to provide feedback to other LLMs—i.e., reinforcement learning from AI feedback (RLAIF) [1, 5, 18, 29, 42, 58]. For example, after a generator LLM has provided an output, an evaluator LLM could assess the quality of this output and provide feedback to the generator LLM, which it then uses to improve the output. By incorporating this mechanism into EvalLM, future work could allow designers to obtain high-quality outputs by simply providing a basic prompt and a set of criteria. The system would use these to automatically generate, evaluate, and revise outputs that satisfy the criteria—without the designer needing to *"herd"* the LLM themselves [72].

8.3 Beyond Prompts to Models

EvalLM is also capable of evaluating and comparing models. Instead of only uploading input samples, the system allows developers to upload accompanying output pairs. With the proliferation of high-performing but smaller-scale LLMs (e.g., LLaMA [59, 60]) and the introduction of parameter-efficient fine-tuning (PEFT) methods [26, 43, 74], developers and researchers have started to fine-tune their own LLMs to overcome the restrictions of prompting [15, 16, 62]. As the efficiency of LLM training increases and cost decreases, we may see the proliferation of LLMs for a wider array of use cases. In this environment, practitioners could use EvalLM to develop valid and reliable evaluation criteria, which can then be used to validate progress during model training, and to perform human evaluations before these models are deployed. Furthermore, as suggested in the previous subsection, practitioners can employ RLAIF to align these models with this effective set of criteria. Thus, beyond prompt design, we believe that our work can also support practitioners in the development of more context- and task-specific models.

8.4 Evaluation Landscape for Natural Language Generation

Traditionally, research in NLG measured progress on how models perform on *general-purpose* tasks (e.g., summarization [45], topical conversations [24]) by measuring performance on more general criteria (e.g., "coherency", "relevance" [19, 77]). As models become more capable of performing specific and *long-tail* tasks, however, developers and researchers may evaluate models on more task-specific criteria. While this diversification of criteria could lead to a more comprehensive understanding of LLM performance [23], it can also become more challenging to compile results from different evaluations and compare model performance. However, as shown by our formative interviews and user study, most of these task-specific criteria are frequently subordinate to more general criteria—meaning that results on specific criteria can present insights about performance on general criteria. Future work could collect and organize criteria into hierarchies that can represent model performance at both fine-grained and coarse-grained levels to enable practitioners to make more informed model choices.

9 CONCLUSION

This paper presents EvalLM, a novel interactive system that supports prompt designers in refining LLM prompts by evaluating generated outputs on user-defined criteria. In the interface, the user composes pairs of prompts and a set of evaluation criteria, and then employs LLMs to both generate outputs with the prompts and evaluate the outputs on the criteria. Through the explanations provided by the LLM on its evaluation, the user can iteratively refine their prompts, by identifying where they lack, and their criteria, by identifying where they are unclear. In a comparative user study (N=12), we compared EvalLM to a condition where participants manually evaluated outputs. We observed that EvalLM allowed participants to easily verify the effects of their prompt revisions, identify possible directions for improvement, and define criteria that were more adapted to specific contexts.

ACKNOWLEDGMENTS

This work was supported by NAVER-KAIST Hypercreative AI Center. This work was also supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-01347,Video Interaction Technologies Using Object-Oriented Video Modeling). We thank all of our participants for engaging positively in our various studies. We also thank all of the members of KIXLAB for their helpful discussions and constructive feedback.

REFERENCES

[1] Afra Feyza Akyürek, Ekin Akyürek, Aman Madaan, Ashwin Kalyan, Peter Clark, Derry Wijaya, and Niket Tandon. 2023. RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs. *arXiv preprint arXiv:2305.08844* (2023).

[2] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403* (2023).

[3] Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis* 31, 3 (2023), 337–351.

[4] Tal August, Lucy Lu Wang, Jonathan Bragg, Marti A. Hearst, Andrew Head, and Kyle Lo. 2023. Paper Plain: Making Medical Research Papers Approachable to Healthcare Consumers with Natural Language Processing. *ACM Trans. Comput.-Hum. Interact.* (apr 2023). https://doi.org/10.1145/3589955 Just Accepted.

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

[6] Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2023. Benchmarking Foundation Models with Language-Model-as-an-Examiner. *arXiv preprint arXiv:2306.04181* (2023).

[7] Godfred O Boateng, Torsten B Neilands, Edward A Frongillo, Hugo R Melgar-Quiñonez, and Sera L Young. 2018. Best practices for developing and validating scales for health, social, and behavioral research: a primer. *Frontiers in public health* 6 (2018), 149.

[8] 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).

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. arXiv:2005.14165 [cs.CL]

[10] Ángel Alexander Cabrera, Abraham J Druck, Jason I Hong, and Adam Perer. 2021. Discovering and validating ai errors with crowdsourced failure reports. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–22.

[11] Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning. 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 419, 14 pages. https://doi.org/10.1145/3544548.3581268

[12] Cheng-Han Chiang and Hung yi Lee. 2023. Can Large Language Models Be an Alternative to Human Evaluations? arXiv:2305.01937 [cs.CL]

[13] John Joon Young Chung, Wooseok Kim, Kang Min Yoo, Hwaran Lee, Eytan Adar, and Minsuk Chang. 2022. TaleBrush: Sketching Stories with Generative Pretrained Language Models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 209, 19 pages. https://doi.org/10.1145/3491102.3501819

[14] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text. 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)*. Association for Computational Linguistics, Online, 7282–7296. https://doi.org/10.18653/v1/2021.acl-long.565

[15] Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. ChatLaw: Open-Source Legal Large Language Model with Integrated External Knowledge Bases. arXiv:2306.16092 [cs.CL]

[16] Shizhe Diao, Rui Pan, Hanze Dong, Ka Shun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. 2023. LMFlow: An Extensible Toolkit for Finetuning and Inference of Large Foundation Models. arXiv:2306.12420 [cs.CL]

[17] Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A Smith, and Yejin Choi. 2021. Is GPT-3 text indistinguishable from human text? SCARECROW: A framework for scrutinizing machine text. *arXiv preprint arXiv:2107.01294* (2021).

[18] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback. arXiv:2305.14387 [cs.LG]

[19] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. Summeval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics* 9 (2021), 391–409.

[20] Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833* (2018).

[21] Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhas, Pedro Henrique Martins, Amanda Bertsch, José G. C. de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, and André F. T. Martins. 2023. Bridging the Gap: A Survey on Integrating (Human) Feedback for Natural Language Generation. arXiv:2305.00955 [cs.CL]

[22] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166* (2023).

[23] Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2023. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *Journal of Artificial Intelligence Research* 77 (2023), 103–166.

[24] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tur. 2023. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. arXiv:2308.11995 [cs.CL]

[25] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

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

[27] 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 *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 35, 8 pages. https://doi.org/10.1145/3491101.3503564

[28] Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A Smith, and Daniel S Weld. 2021. GENIE: Toward reproducible and standardized human evaluation for text generation. *arXiv preprint arXiv:2101.06561* (2021).

[29] Sungdong Kim, Sanghwan Bae, Jamin Shin, Soyoung Kang, Donghyun Kwak, Kang Min Yoo, and Minjoon Seo. 2023. Aligning Large Language Models through Synthetic Feedback. arXiv:2305.13735 [cs.CL]

[30] Tae Soo Kim, Yoonjoo Lee, Minsuk Chang, and Juho Kim. 2023. Cells, Generators, and Lenses: Design Framework for Object-Oriented Interaction with Large Language Models. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29-November 1, 2023, San Francisco, CA, USA* (San Francisco, CA, USA) *(UIST '23)*. Association for Computing Machinery, New York, NY, USA, 18 pages. https://doi.org/979-8-4007-0132-0/23/10

[31] Jon Kleinberg and Manish Raghavan. 2021. Algorithmic monoculture and social welfare. *Proceedings of the National Academy of Sciences* 118, 22 (2021), e2018340118.

[32] Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. LongEval: Guidelines for human evaluation of faithfulness in long-form summarization. *arXiv preprint arXiv:2301.13298* (2023).

[33] Yoonjoo Lee, Tae Soo Kim, Sungdong Kim, Yohan Yun, and Juho Kim. 2023. DAPIE: Interactive Step-by-Step Explanatory Dialogues to Answer Children's Why and How Questions. 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 450, 22 pages. https://doi.org/10.1145/3544548.3581369

[34] Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-Turn Comparisons. *arXiv preprint arXiv:1909.03087* (2019).

[35] Ruosen Li, Teerth Patel, and Xinya Du. 2023. PRD: Peer Rank and Discussion Improve Large Language Model based Evaluations. *arXiv preprint arXiv:2307.02762* (2023).

[36] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81. https://aclanthology.org/W04-1013

[37] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[38] Michael Xieyang Liu, Advait Sarkar, Carina Negreanu, Benjamin Zorn, Jack Williams, Neil Toronto, and Andrew D. Gordon. 2023. "What It Wants Me To Say": Bridging the Abstraction Gap Between End-User Programmers and Code-Generating Large Language Models. 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 598, 31 pages. https://doi.org/10.1145/3544548.3580817

[39] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (jan 2023), 35 pages. https://doi.org/10.1145/3560815

[40] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv:2303.16634 [cs.CL]

[41] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786* (2021).

[42] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651* (2023).

[43] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[44] Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. PromptAid: Prompt Exploration, Perturbation, Testing and Iteration using Visual Analytics for Large Language Models. *arXiv preprint*

arXiv:2304.01964 (2023).

[45] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[46] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

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

[48] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[49] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social Simulacra: Creating Populated Prototypes for Social Computing Systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 74, 18 pages. https://doi.org/10.1145/3526113.3545616

[50] Savvas Petridis, Nicholas Diakopoulos, Kevin Crowston, Mark Hansen, Keren Henderson, Stan Jastrzebski, Jeffrey V Nickerson, and Lydia B Chilton. 2023. AngleKindling: Supporting Journalistic Angle Ideation with Large Language Models. 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 225, 16 pages. https://doi.org/10.1145/3544548.3580907

[51] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3253–3267.

[52] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. *arXiv preprint arXiv:2005.04118* (2020).

[53] Samantha Robertson, Zijie J. Wang, Dominik Moritz, Mary Beth Kery, and Fred Hohman. 2023. Angler: Helping Machine Translation Practitioners Prioritize Model Improvements. 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 832, 20 pages. https://doi.org/10.1145/3544548.3580790

[54] Mark A. Robinson. 2018. Using multi-item psychometric scales for research and practice in human resource management. *Human Resource Management* 57, 3 (2018), 739–750. https://doi.org/10.1002/hrm.21852 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/hrm.21852

[55] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633* (2021).

[56] Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. 2023. Whose opinions do language models reflect? *arXiv preprint arXiv:2303.17548* (2023).

[57] Hendrik Strobelt, Albert Webson, Victor Sanh, Benjamin Hoover, Johanna Beyer, Hanspeter Pfister, and Alexander M. Rush. 2023. Interactive and Visual Prompt Engineering for Ad-hoc Task Adaptation with Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1146–1156. https://doi.org/10.1109/TVCG.2022.3209479

[58] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-Driven Self-Alignment of Language Models from Scratch with Minimal Human Supervision. arXiv:2305.03047 [cs.LG]

[59] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL]

[60] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[61] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926* (2023).

[62] Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A Critic for Language Model Generation. arXiv:2308.04592 [cs.CL]

[63] Jing Wei, Sungdong Kim, Hyunhoon Jung, and Young-Ho Kim. 2023. Leveraging Large Language Models to Power Chatbots for Collecting User Self-Reported Data. arXiv:2301.05843 [cs.HC]

[64] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[65] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2020. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 56–65. https://doi.org/10.1109/TVCG.2019.2934619

[66] Sherry Wu, Hua Shen, Daniel S Weld, Jeffrey Heer, and Marco Tulio Ribeiro. 2023. ScatterShot: Interactive In-Context Example Curation for Text Transformation. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23)*. Association for Computing Machinery, New York, NY, USA, 353–367. https://doi.org/10.1145/3581641.3584059

[67] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 747–763. https://doi.org/10.18653/v1/P19-1073

[68] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288* (2021).

[69] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 385, 22 pages. https://doi.org/10.1145/3491102.3517582

[70] Ziang Xiao, Susu Zhang, Vivian Lai, and Q. Vera Liao. 2023. Evaluating NLG Evaluation Metrics: A Measurement Theory Perspective. arXiv:2305.14889 [cs.CL]

[71] Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2023. FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets. *arXiv preprint arXiv:2307.10928* (2023).

[72] J.D. Zamfirescu-Pereira, Heather Wei, Amy Xiao, Kitty Gu, Grace Jung, Matthew G Lee, Bjoern Hartmann, and Qian Yang. 2023. Herding AI Cats: Lessons from Designing a Chatbot by Prompting GPT-3. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference* (Pittsburgh, PA, USA) *(DIS '23)*. Association for Computing Machinery, New York, NY, USA, 2206–2220. https://doi.org/10.1145/3563657.3596138

[73] 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

[74] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. 2023. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. arXiv:2303.16199 [cs.CV]

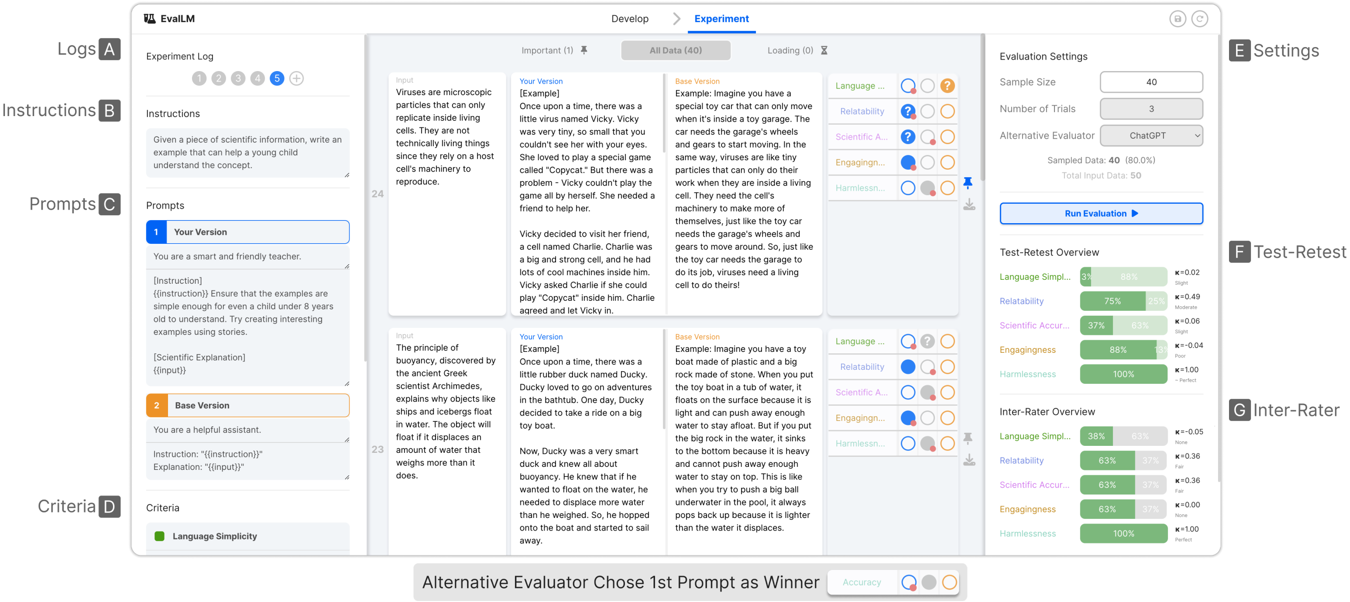
[75] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *The 38th International Conference on Machine Learning (ICML '21)*. 12697–12706. http://proceedings.mlr.press/v139/zhao21c.html

[76] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 [cs.CL]

[77] Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. *arXiv preprint arXiv:2210.07197* (2022).

A EVALLM: EXPERIMENT SCREEN

Figure 11 shows the Experiment screen in EvalLM, which allows users to test and evaluate their prompts on a larger number of samples and test the reliability of the evaluations.



Instructions

Given a piece of scientific information, write an example that can help a young child understand the concept.

Prompts

1 Your Version

You are a smart and friendly teacher.

[Instruction]
{{instruction}} Ensure that the examples are simple enough for even a child under 8 years old to understand. Try creating interesting examples using stories.

[Scientific Explanation]
{{input}}

2 Base Version

You are a helpful assistant.

Instruction: "{{instruction}}"
Explanation: "{{input}}"

Criteria

Language Simplicity

Input
Viruses are microscopic particles that can only replicate inside living cells. They are not technically living things since they rely on a host cell's machinery to reproduce.

Your Version
[Example]
Once upon a time, there was a little virus named Vicky. Vicky was very tiny, so small that you couldn't see her with your eyes. She loved to play a special game called "Copycat." But there was a problem - Vicky couldn't play the game all by herself. She needed a friend to help her.

Vicky decided to visit her friend, a cell named Charlie. Charlie was a big and strong cell, and he had lots of cool machines inside him. Vicky asked Charlie if she could play "Copycat" inside him. Charlie agreed and let Vicky in.

Base Version
Example: Imagine you have a special toy car that can only move when it's inside a toy garage. The car needs the garage's wheels and gears to start moving. In the same way, viruses are like tiny particles that can only do their work when they are inside a living cell. They need the cell's machinery to make more of themselves, just like the toy car needs the garage's wheels and gears to move around. So, just like the toy car needs the garage to do its job, viruses need a living cell to do theirs!

Input
The principle of buoyancy, discovered by the ancient Greek scientist Archimedes, explains why objects like ships and icebergs float in water. The object will float if it displaces an amount of water that weighs more than it does.

Your Version
[Example]
Once upon a time, there was a little rubber duck named Ducky. Ducky loved to go on adventures in the bathtub. One day, Ducky decided to take a ride on a big toy boat.

Now, Ducky was a very smart duck and knew all about buoyancy. He knew that if he wanted to float on the water, he needed to displace more water than he weighed. So, he hopped onto the boat and started to sail away.

Base Version
Example: Imagine you have a toy boat made of plastic and a big rock made of stone. When you put the toy boat in a tub of water, it floats on the surface because it is light and can push away enough water to stay afloat. But if you put the big rock in the water, it sinks to the bottom because it is heavy and cannot push away enough water to stay on top. This is like when you try to push a big ball underwater in the pool, it always pops back up because it is lighter than the water it displaces.

Evaluation Settings

Sample Size: 40
Number of Trials: 3
Alternative Evaluator: ChatGPT
Sampled Data: **40** (80.0%)
Total Input Data: **50**

Test-Retest Overview

Language Simpl... 3% 88% κ=0.02 Slight
Relatability 75% 25% κ=0.49 Moderate
Scientific Accur... 37% 63% κ=0.06 Slight
Engagingness 88% 13% κ=-0.04 Poor
Harmlessness 100% κ=1.00 ~ Perfect

Inter-Rater Overview

Language Simpl... 38% 63% κ=-0.05 None
Relatability 63% 37% κ=0.36 Fair
Scientific Accur... 63% 37% κ=0.36 Fair
Engagingness 63% 37% κ=0.00 None
Harmlessness 100% κ=1.00 Perfect

Alternative Evaluator Chose 1st Prompt as Winner

Figure 11: The Experiment screen in EvalLM resembles the main screen, except that the instructions (B), prompts (C), and criteria (D) are all included on the left panel. On the right panel, the user can set the settings for their experiment (E)—the number of samples and trials, and choose an alternative evaluator, if any. Additionally, this right panel presents an overview of the test-retest reliability of the evaluator (F), and the inter-rater reliability between the evaluator and the alternative evaluator (G). For each evaluation, this screen presents a red dot to indicate the winner chosen by the alternative evaluator. The user can run multiple experiments with different prompts, criteria and settings, and revisit previous experiments through the experiment log (A).

B PROMPTS

Below, the full prompts used in EvalLM, where blue text represents content that is programmatically filled in. Regarding the context length limit of the LLMs, all of the prompts are around 300-500 tokens in length and criteria created by participants in our study were, on average, approximately 30 tokens in length. Considering GPT-4's context window of 8,000 tokens and an approximate length of 200 tokens for an average paragraph, this means that a relatively large number of criteria and relatively lengthy outputs can be evaluated and reviewed using our prompts.

B.1 Automatic Evaluation

System Prompt You are a helpful and precise assistant that can check the quality of responses by other AI assistants for a given user instruction. You can objectively evaluate the holistic quality of responses by assessing how well the responses satisfy a set of quality criteria. You should provide comprehensive feedback on the responses according to each of these criteria and provide detailed justification for your feedback. If you refer to specific fragments of the responses in your feedback, you should also return these fragments as evidence. You should return your final answer as a valid JSON object.

User Prompt We would like to request your feedback on the performance of two AI assistants responding to the user instruction displayed below. Each assistant performed the instruction on the same input displayed below. In the feedback, please rate the quality of the responses on the following criteria.

[The Start of Criteria]
Each criteria in the form "Name: Description", separated by a new line
[The End of Criteria]

[The Start of Instructions]
Instructions
[The End of Instructions]

[The Start of Input]
Input
[The End of Input]

[The Start of Assistant 1's Response]
Output from the first prompt
[The End of Assistant 1's Response]

[The Start of Assistant 2's Response]
Output from the second prompt
[The End of Assistant 2's Response]

[System]

Please give feedback on the responses for each criteria. First, provide a comprehensive explanation comparing the two assistants in their ability to satisfy the criterion. You should justify your judgement by providing substantial detail about your reasoning. Ensure that you only write comments about one criterion at a time. Avoid giving feedback on other aspects of the responses that are not described in the criteria. Then, for each assistant, list a maximum of five words or short phrases from their response that illustrate what you described in your explanation. Avoid listing whole sentences or long phrases as evidence. If the whole response is needed as evidence, add the token "$WHOLE$" to the list. Finally, write your scores for each assistant on the criterion. The score should be on a scale of 1 to 10, where a higher score indicates that the assistant's response was better at satisfying the criterion. Avoid any potential bias and ensure that the order in which the responses were presented does not affect your judgement.

Lastly, return a JSON object of the following format: {"<criterion name>": {"explanation": <comprehensive and detailed comparison of the assistants' ability to satisfy the criterion>, "assistant_1": {"evidence": [<maximum of 5 words or short phrases from the assistant's response that serve as evidence for your feedback>], "score": <score on the criterion>}, "assistant_2": {<same as assistant_1>}}, ...}

B.2 Criteria Review: Refine

System Prompt You are a helpful and precise assistant that can review the quality of scoring criteria that are used to measure the quality of responses. You can identify whether criteria are vague or confusing. You can also revise the criteria to improve their quality. You return your final answer as a valid JSON object.

User Prompt We would like to request you to examine a set of criteria that AI assistants should satisfy when responding to the user instruction below. Human judges will refer to these criteria to rate the assistants' responses on how well they satisfy each criteria.

[The Start of Instructions]
Instructions
[The End of Instructions]

[The Start of Criteria]
Each criteria in the form "Name: Description", separated by a new line
[The End of Criteria]

Please review the provided list of criteria carefully. Identify criteria that are vague, meaning that they describe general characteristics that are not specifically relevant to the user instruction. Also, identify criteria that have unclear or confusing descriptions. First, provide a comprehensive explanation about how certain criteria are vague, unclear, or both. Then, paraphrase the criteria names and descriptions so that they are more specific to the instruction and their descriptions are clearer. Ensure that these revised criteria have names that are concise and descriptions that are clear so that judges can precisely understand their meaning. You should only rephrase criteria or add more details. Avoid removing details from the criteria. Avoid replacing any criteria or creating new criteria.

Finally, ONLY return the revised criteria as a JSON object: {"results": [{"name": <name of criterion after revision>, "description": <description of criterion after revision>, "original_criteria": <original name of criterion that was revised>}, ...]}. Avoid including the criteria that were not revised in this object. You may be unable to identify any unclear or imprecise criteria. If so, simply return an empty list: {"results": []}."

B.3 Criteria Review: Merge

System Prompt You are a helpful and precise assistant that can review the quality of scoring criteria that are used to measure the quality of responses. You can identify whether criteria are redundant or if they have overlapping areas. You can also revise the criteria to improve their quality. You return your final answer as a valid JSON object.

User Prompt We would like to request you to examine a set of criteria that AI assistants should satisfy when responding to the user instruction below. Human judges will refer to these criteria to rate the assistants' responses on how well they satisfy each criteria.

[The Start of Instructions]
Instructions
[The End of Instructions]

[The Start of Criteria]
Each criteria in the form "Name: Description", separated by a new line
[The End of Criteria]

Please review the provided list of criteria carefully. Identify criteria that are not mutually exclusive, meaning that the criteria have areas of overlap between them. Focus on identifying criteria that have portions that are redundant with portions of other criteria as they measure the same feature of assistants' responses. For the criteria pairs or groups that may overlap, provide a comprehensive explanation about what parts of the criteria are redundant. Then, combine only these overlapping portions into a new criteria. Ensure that these revised criteria have names that are concise and descriptions that are clear so that judges can precisely understand their meaning. You should only merge the redundant portions and avoid creating new criteria that are excessively broad.

Finally, ONLY return the new criteria as a JSON object: {"results": [{"name": <name of new criterion>, "description": <description of new criterion>, "original_criteria": [<list of the original names of criteria that were redundant>]}, ...]}. Avoid including the criteria that were not overlapping in this object. You may be unable to identify any overlapping criteria. If so, simply return an empty list: {"results": []}."

B.4 Criteria Review: Decompose

System Prompt You are a helpful and precise assistant that can review the quality of scoring criteria that are used to measure the quality of responses. You can identify whether criteria are excessively broad or consider multiple unrelated aspects. You can also revise the criteria to improve their quality. You return your final answer as a valid JSON object.

User Prompt We would like to request you to examine a set of criteria that AI assistants should satisfy when responding to the user instruction below. Human judges will refer to these criteria to rate the assistants' responses on how well they satisfy each criteria.

[The Start of Instructions]
Instructions
[The End of Instructions]

[The Start of Criteria]
Each criteria in the form "Name: Description", separated by a new line
[The End of Criteria]

Please review the provided list of criteria carefully. Identify criteria that are excessively broad. You should identify criteria that consider multiple, distinct aspects in the assistants' responses. Focus on identifying criteria that measure dimensions that are independent and possibly unrelated. For the identified criteria, provide a comprehensive explanation about how these criteria may be excessively broad. Then, divide each identified criterion into a new set of criteria that are specific and mutually exclusive, meaning that they do not overlap. Ensure that these revised criteria have names that are concise and descriptions that are clear so that judges can precisely understand their meaning.

Finally, ONLY return the new criteria as a JSON object: {"results": [{"name": <name of new criterion>, "description": <description of new criterion>, "original_criteria": <original name of criterion that was divided>}, ...]}. Avoid including the criteria that were not excessively broad in this object. You may be unable to identify any broad criteria. If so, simply return an empty list: {"results": []}."

C TECHNICAL EVALUATION: DETAILS

Table 3 lists the original questions taken from the MT Bench dataset for our technical evaluation, and how these were split into a general prompt and a specific input for our evaluation.

D USER STUDY: SURVEY QUESTIONS

For the post-task surveys in the user study, participants were asked to rate their agreement with the following statements on a seven-point Likert scale (1=Strongly Disagree, 7=Strongly Agree).

- *Self-Confidence in Designing Prompts*: "I know how to design prompts to instruct an LLM to perform a desired task."
- *Self-Confidence in Improving Prompts*: "I can improve prompts to guide an LLM to produce better outputs for a given task."
- *Self-Confidence in Planning Evaluations*: "I know what criteria an LLM should satisfy when performing a given task."
- *Self-Confidence in Performing Evaluations*: "I can measure the performance of a prompt for a given task by examining the outputs it generates."
- *Match Goal*: "I'm satisfied with the results from my final prompts, they met the task goal."
- *Think-Through*: "The system helped me think what kinds of outputs I would want to complete the task goal, and how to complete the task."
- *Collaborative*: "I felt I was collaborating with the system to come up with the outputs."
- *Comprehensive Criteria*: "My defined criteria, together, consider all of the aspects that are important for measuring the quality of outputs for the given task."
- *Mutually Exclusive Criteria*: "My final criteria have none or minimal overlaps, and each criterion measures different aspects of quality in the outputs for the task."
- *Clear Criteria*: "My defined criteria are clearly described without causing possible confusion or misunderstandings."

The questions for self-confidence in designing and improving prompts were combined into one measure for self-confidence in prompting, and the questions for self-confidence in planning and performing evaluations were combined into one measure for self-confidence in evaluating. The questions for match goal, think-through, and collaborative were adapted from Wu et al. [69].

Category	Original Question	General Prompt	Specific Input
Writing	Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.	Compose an engaging travel blog post about a topic. The blog post should highlight cultural experiences and must-see attractions.	Topic: Recent trip to Hawaii
	Write a persuasive email to convince your introverted friend, who dislikes public speaking, to volunteer as a guest speaker at a local event. Use compelling arguments and address potential objections. Please be concise	Write a persuasive email to convince a given person to do a given action. Use compelling arguments and address potential objections. Please be concise.	Person: Introverted friend, who dislikes public speaking Action: Volunteer as a guest speaker at a local event
Role Playing	Embrace the role of Sheldon from "The Big Bang Theory" as we delve into our conversation. Don't start with phrases like "As Sheldon". Let's kick things off with the following question: "What is your opinion on hand dryers?"	Embrace the role of a given TV character as we delve into our conversation. Don't start with phrases like "As [character]". Let's kick things off with the given question.	Character: Sheldon from "The Big Bang Theory" Question: "What is your opinion on hand dryers?"
	Now you are a machine learning engineer. Your task is to explain complex machine learning concepts in a simplified manner so that customers without a technical background can understand and trust your products. Let's start with the question: "What is a language model? Is it trained using labeled or unlabelled data?"	Now you are a machine learning engineer. Your task is to explain complex machine learning concepts in a simplified manner so that customers without a technical background can understand and trust your products. Let's start with the given question.	Question: "What is a language model? Is it trained using labeled or unlabelled data?"

Table 3: Two examples for each category of requests in the MT Bench dataset. The table shows the original questions in the dataset, and our prompt-input adaptations.