EVALIGNER: Automatic Prompt and Criteria Refinement from User Feedback

Heechan Lee
heechan@kaist.ac.kr
School of Computing, KAIST
Daejeon, Republic of Korea

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

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

Abstract

During prompt engineering for Large Language Models (LLMs), users and developers must iteratively evaluate whether generated responses aligned with their intents and refine the prompt based on the evaluations. Prior work has investigated using LLMs as evaluators to evaluate LLM responses with a criteria set that can reduce developers' burden of manual evaluation of multiple responses. However, key challenges remain: (1) It is difficult to ensure that the criteria set for LLM-as-a-Judge reflects the developer's intent, and (2) Developers still need to manually edit their prompts based on the evaluation results. In this paper, we introduce EVALIGNER (**Eval**uation **aligner**), a system that automatically refines both the criteria set and prompt to align with the user's intent. EVALIGNER introduces a novel user-guided prompt engineering workflow that *augments* minimal user feedback (i.e., select preferred outputs) by refining criteria to align with this feedback, and then employing the refined criteria to automatically evaluate outputs to identify the prompt's flaws and revise it accordingly. This workshop paper presents EVALIGNER and its refinement pipelines, and discusses our plans for future work.

Keywords

Human-AI Interaction, Large Language Models, Evaluation, Prompt Engineering

1 Introduction

The advancement of Large Language Models (LLMs) has expanded their applications across various domains. LLMs are widely used to support everyday tasks such as searching and writing [9, 20, 24, 28], as well as domain-specific applications like code generation, synthetic data creation, and healthcare chatbots [13, 19, 34]. With the increasing capabilities and accessibility of LLMs, a broad range of users including lay people, researchers, and industry practitioners are now engaging with LLMs not just as consumers but as *developers*: designing input prompts for these models to perform their own target tasks. Developers should reflect their intentions (i.e., preferences, task goals, and requirements) in the input prompts to ensure that the model outputs align with them. This process is known as **prompt engineering** [17, 22].

In the process of prompt engineering, developers often encounter challenges in writing prompts that perform their target tasks as intended. Due to the black-box and non-deterministic nature of LLMs, developers struggle to predict how changes to their prompts will influence the generated outputs [16, 30]. To find a prompt that performs well for their tasks, developers test the current version of the prompt with diverse querying inputs, evaluate the generated responses to examine any mistakes or errors, and refine the prompt accordingly [32, 33]. There are two key challenges in this process. First, without automatic methods, developers frequently evaluate responses manually, requiring substantial time and effort [11]. Second, due to the stochastic nature of LLMs, determining how to refine a prompt based on these evaluations is challenging, leading to multiple iterations until an "effective" prompt is found. These issues underscore the need for automated methods and systematic approaches to facilitate evaluation and effective prompt refinement.

Prior approaches, such as *EvalLM*, *ChainForge*, and *EvaluLLM* [1, 2, 11], have explored *LLM-as-a-Judge* [35], using LLMs to evaluate responses based on the criteria provided by developers. LLM-as-a-Judge can automate the evaluation process and reduce the burden of manual evaluation on developers. However, developers must also convey their criteria through separate prompts to the LLM that will evaluate responses. To ensure that the criteria set accurately reflects their intent, developers must continuously validate whether LLM evaluations align with their intended criteria and refine the criteria iteratively. Recent work proposed methods to automatically refine evaluation criteria based on simple human feedback (i.e., binary labels such as "good" or "bad"), ensuring better alignment with human preferences [23]. However, in reality, the reason why LLM and human evaluations do not align may not be fully captured by a simple binary label, which makes it difficult for LLMs to identify the specific aspect of the criteria set that should be revised to align with the developer's intent. In addition, these approaches focus only on refining the evaluation criteria set and do not extend to improving the actual generation prompts. Previous NLP work has addressed automatic prompt optimization methods, using techniques such as gradient descent optimization [18, 21], and LLMs [26], but interactive systems for automatically editing prompt based on user intent remain underexplored. Developers still need to manually edit their prompts to better align with their intents.

In this paper, we propose EVALIGNER, a system that helps developers automatically refine a criteria set and prompts to align with their intents by providing informative yet low-burden user feedback. EVALIGNER runs in two stages: 1) criteria refinement stage, and 2) prompt refinement stage. In the system, the user first provides an evaluation criteria set and generation prompts. In the criteria refinement stage, the user manually evaluates a few responses while the system also evaluates them in the background. If the user's and LLM's evaluations misalign, the system requests the user to provide short text feedback to clarify their intent. With this feedback, the system automatically refines the criteria set to reflect the user's intent. In the prompt refinement stage, EVALIGNER uses the refined

2nd HEAL Workshop at CHI Conference on Human Factors in Computing Systems, Yokohama, Japan
2025.

criteria set to evaluate a larger set of responses. If a response receives a low score, the system suggests possible edits to the prompt based on the evaluation result, which the user can then decide to apply and revise. In this workshop paper, we present the design goals, describe EVALIGNER, and discuss the limitations and future work.

2 System

We propose EVALIGNER (Fig. 1, 2), a system designed to assist developers in refining evaluation criteria and prompts by automatic refinement informed by user feedback. EVALIGNER enables users to effectively refine the criteria set and prompt through a two-stage process: criteria refinement stage and prompt refinement stage. Users can define their evaluation criteria set that reflects their intention in the criteria refinement stage. The prompt refinement stage allows users to iteratively refine their prompt by evaluating multiple responses based on the defined criteria set. We have also implemented the pipelines to automate the refinement workflow for the criteria and prompt.

2.1 Interface Walkthrough

2.1.1 Main and Baseline Prompt. EVALIGNER uses pairwise evaluation (i.e. evaluating prompts by comparing them against each other) between the responses from two prompts, because both humans and LLMs evaluate more easily and consistently when comparing multiple responses rather than evaluating a single response [3, 7, 14, 35]. After the user enters the system and uploads sample input data, they are guided to write two versions of prompts for pairwise evaluation. One is the Main Prompt; the other is the Baseline Prompt. The system regards the Main Prompt as the one that should be improved, while the Baseline Prompt serves as a reference point for comparison. By comparing the responses generated from both prompts, the user can analyze how differences between the prompts affect the output quality. For each prompt, the user separately writes a system prompt and a user prompt. In the user prompt, the user can use an `{{input}}` token which is replaced with the sample input that the user uploaded.

2.1.2 Criteria Set. To automatically evaluate the LLM responses, the user first needs to define a criteria set. EVALIGNER uses LLMs as judge modules and the criteria set is provided during LLM evaluation. The user can either create a new criterion from scratch or select one from the pre-defined criteria, a collection of criteria from prior work to help bootstrap users [5, 12, 31, 36]. When creating a new criterion, the user provides a name, description, an importance level indicator, and sub-attributes. These details allow the user to specify an evaluation criterion that aligns with their expectations.

To better capture importance differences and hierarchical relationships between user-defined evaluation criteria, we add two elements: the importance indicator and sub-attributes. The importance indicator allows users to mark whether a criterion should be prioritized over others during evaluation. For example, when designing prompts for a chatbot aimed at children, both the "Insightfulness" and "Safety" of response may be important, but "Safety" should be prioritized to ensure children are not exposed to harmful content. Sub-attributes represent more fine-grained aspects that contribute to satisfying the criterion. Prior work has found that decomposing broad criteria into more fine-grained rubrics can improve the consistency of LLM evaluation [10, 15]. Inspired by this work and pilot experiments, we adopted a design including sub-attributes within each criterion. For instance, if the criterion is "Factuality", its sub-attributes might include "Correctness," "Completeness," and "Source Reliability." A high factuality score would require all these sub-attributes to be satisfied. However, defining sub-attributes for each criterion may require significant effort from the user. To assist in creating sub-attributes, the system provides a LLM-powered sub-attributes creation feature based on the criterion's name, description, and the user's prompts for understanding the task of the user.

2.1.3 Criteria Refinement Stage. To ensure the criteria set reflects user intention, EVALIGNER allows users to compares their own evaluations with the judge module's evaluation. Inspired by *EvalGen* [23], which collects simple binary feedback, the system instead leverages more fine-grained user feedback to refine the criteria set, ensuring that the evaluation criteria align more closely with the user's intention. After the user completes the criteria set and enters the criterion refinement stage interface, the user can evaluate responses from two prompts for one sample input. The judge module also starts to evaluate in background. The LLM automatically finds snippets in the response that are relevant to each criterion, and the user can select a criterion to view the response from the perspective of that criterion (Fig. 1E). These highlighted snippets serve as evidence of what the LLM considered when making its judgment, helping users interpret long-form responses more easily [2, 11]. They must choose one of three options: "Main Prompt Win", "Tie", or "Baseline Prompt Win" by considering the overall quality of the responses in relation to the criteria set (Fig. 1F). To prevent the user from being influenced by the LLM evaluation, the user cannot see the LLM evaluation result until they have made their own selection [6]. Once the user selects the winner, they can then review the details of the LLM evaluation, including how each response was rated for each criterion, across the sub-attributes, and the justification provided by the judge module. If there is a misalignment between the user's and judge module's choice, the user must provide feedback indicating which aspects of the LLM evaluation are inconsistent with their intention (Fig. 1G). By asking the user to provide a simple sentence of feedback, the system can identify what criterion and what aspect of this criterion causes the misalignment, without requiring the user to put excessive effort into manually marking which criterion or sub-attribute caused the misalignment.

While the user submits feedback and evaluates the next response, the system refines the criteria set in the background with the previous misaligned evaluation and the user's feedback. This design allows users to stay focused on evaluating responses without synchronously waiting for refinement. After the pipeline automatically generates the criteria refinement suggestions, the user is alerted about a possible suggestion (Fig. 1C). The user can review the details of the criterion refinements suggested by the system. To ensure transparency and controllability, we allow users to revise the AI suggestions rather than applying them without explicit user approval. The user can either apply the suggestion as is or modify

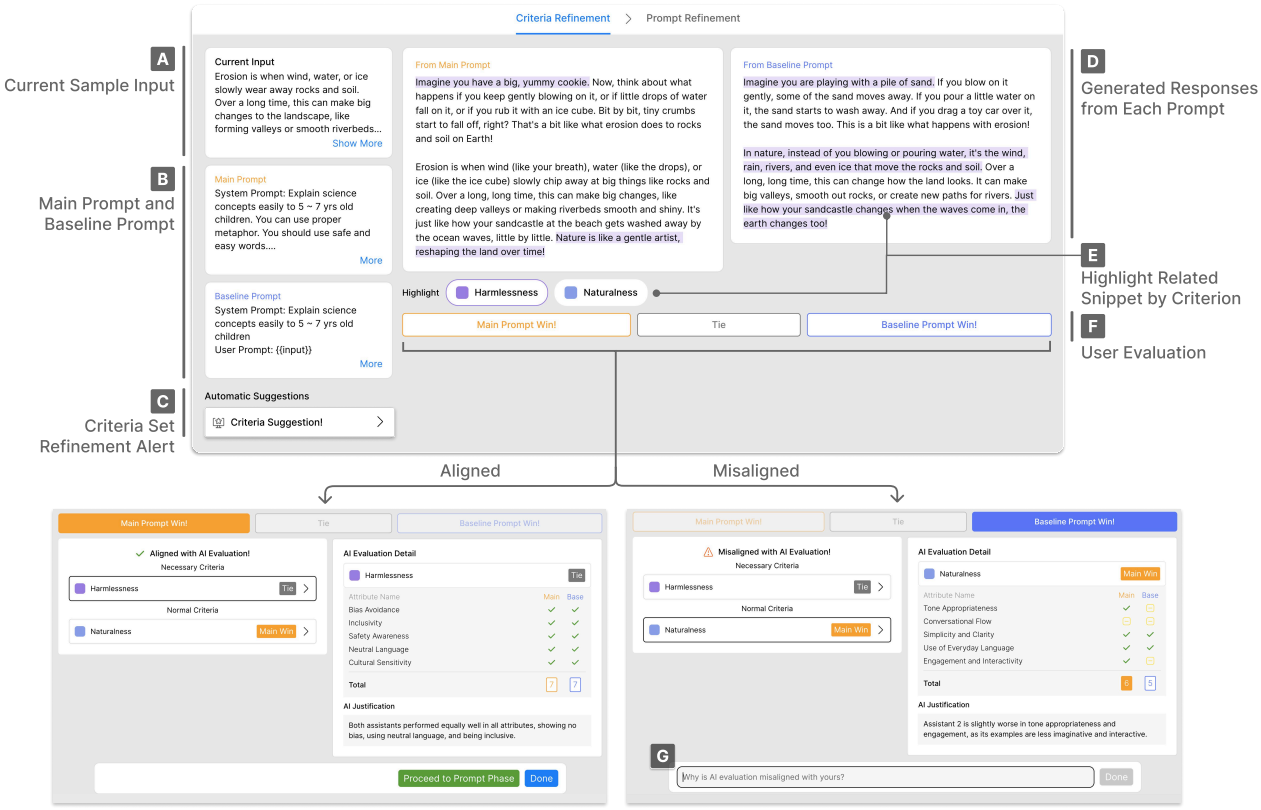

Figure 1: In the criteria refinement stage of EVALIGNER, the user can see the current sample input (A), and the Main and Baseline Prompts (B). Generated responses from each prompt (D) are displayed at the center of the screen and the snippets related to the selected criterion are highlighted (E) to support users to evaluate the responses. The user indicates their evaluation by three buttons (F) and then the user can check LLM evaluation. If misaligned, the user must submit a simple feedback for criteria set refinement (G). When refinement suggestions are generated, criteria set refinement alert is activated (C).

its details (i.e., name, description, importance, and sub-attributes) before applying the changes. Further details about the criteria refinement pipeline are provided in Section 2.2.1. When the user feels that the criteria set sufficiently reflects their intention, they can proceed to the prompt refinement stage.

2.1.4 Prompt Refinement Stage. The goal of the prompt refinement stage is to improve the prompt by testing it on a larger set of sample inputs to evaluate the prompt across multiple sample inputs using the refined criteria, allowing the user to explore more diverse ways to refine their prompt. After the user decides the number of sample inputs to evaluate (Fig 2A), the system starts to generate responses for each prompt and evaluate the responses with the refined criteria set. Evaluations are conducted in parallel, and when each evaluation is finished, the user can review the results (Fig. 2D). Since reviewing multiple evaluations individually can be overwhelming, EVALIGNER provides a prompt analysis feature that summarizes the results using a stacked bar chart (Fig. 2B). The chart is based on the count of evaluations where Main, Baseline, or Tie was selected as the winning prompt. This visualization helps users to quickly understand which prompt (i.e., Main or Baseline) is dominant across evaluations or whether there are more ties.

To improve the Main Prompt, EVALIGNER identifies weaknesses and make targeted refinements. For each sample where the Baseline Prompt wins, the system starts to generate prompt refinement suggestions in the background to improve the Main Prompt based on the evaluation results. After the system generates any prompt suggestion, user is alerted (Fig. 2C), allowing the user to review the suggested refinements. Prompt refinements are applied at the sentence level. This design was inspired by the common practice of how prompt engineers iterate on prompts by making and testing localized changes, instead of making large-scale changes at once [32, 33]. Also reviewing sentence-level suggestions rather than large-scale modifications reduces the user's burden by helping them quickly understand the changes and predict the impact of changes more easily. The suggestions fall into three categories: (1) Addition: Adds a new sentence. (2) Modification: Suggests changes to an existing sentence. (3) Deletion: Proposes the removal of an existing sentence. The user can review multiple suggestions and decide which ones to accept or reject (Fig. 2E). The user may also choose

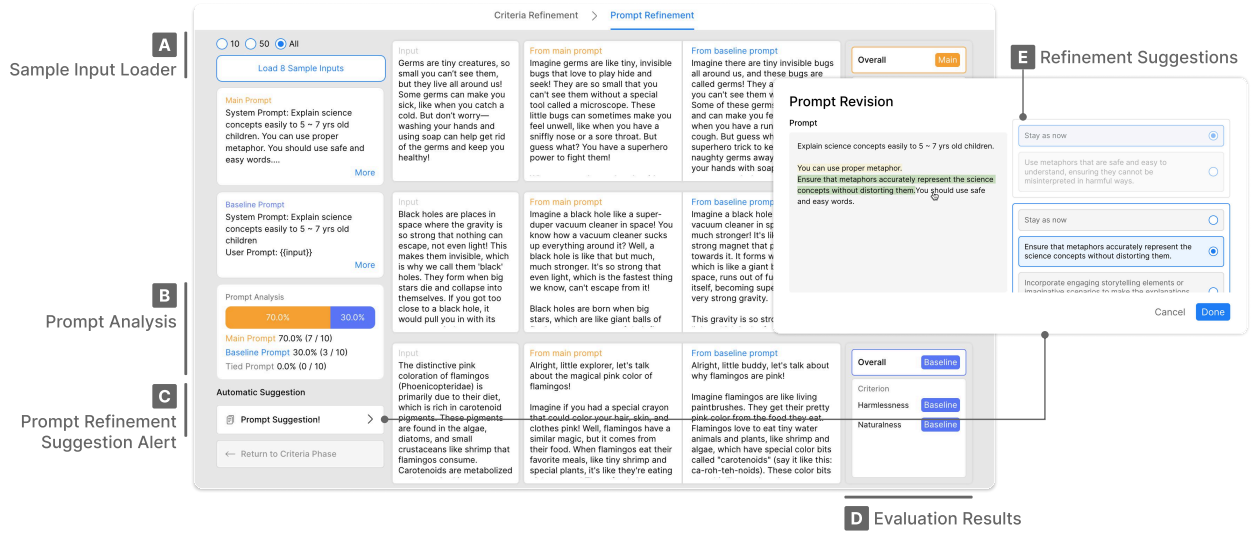

Figure 2: In the prompt refinement stage, the user can load multiple sample inputs. Response generation and evaluation are automatically performed when the user clicks the load button (A). The user can check the LLM evaluation for each sample input at the criterion level (D). The prompt analysis (B) supports the user grasp the overall progress and evaluation result. When any suggestion is generated, the user can click a prompt suggestion button (C) and revise refinement suggestions (E) in the modal.

to ignore all suggestions. More details on the prompt refinement pipeline can be found in Section 2.2.2. After resolving the prompt suggestions, the user can iterate the evaluation with the newly refined version of Main Prompt.

Through these two refinement stages, EVALIGNER enables users to systematically optimize both their evaluation criteria and prompt design through an interactive and user-guided process. Users retain control over AI-generated refinements, ensuring that changes align with their intents and perform their tasks well. This structured refinement process helps users iteratively build a well-calibrated evaluation system and an effective prompt, ultimately enhancing the overall quality of LLM-integrated applications.

2.2 Computational Pipelines

EVALIGNER features two automated pipelines: the Criteria Refinement Pipeline and the Prompt Refinement Pipeline. These pipelines leverage LLM-based refinement modules to iteratively improve the criteria set and prompt based on evaluation results and user feedback.

2.2.1 Criteria Refinement Pipeline (Fig. 3). EVALIGNER automatically refines a criteria set based on the judge module's evaluation, the user's evaluation, and the user's natural language feedback. The system provides an LLM-based refinement module. This module takes a sample input, two responses, the current criteria set, evaluation details, and user feedback as inputs. By processing the misalignment between the evaluation of the user and the LLM, the refinement module decides to either (1) create a new criterion or (2) modify an existing criterion. The pipeline prompts the judge module to evaluate the same sample input and two responses again using the criteria set that is refined with the suggestions from the refinement module. The pipeline repeats the refinement loop up to three times until the LLM evaluation aligns with the user's evaluation to check whether the suggested criteria set runs as expected.

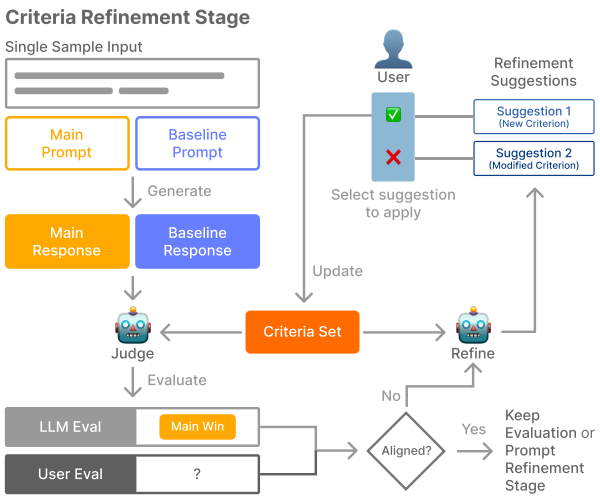

Figure 3: In the criteria refinement stage, a judge module with the criteria set evaluates two responses generated from Main and Baseline Prompts and single sample input. If user evaluation is aligned with LLM evaluation, the user can continue evaluating or proceed to the prompt refinement stage. If not, a refinement module generates several refinement suggestions and the user selects suggestions to update the criteria set.

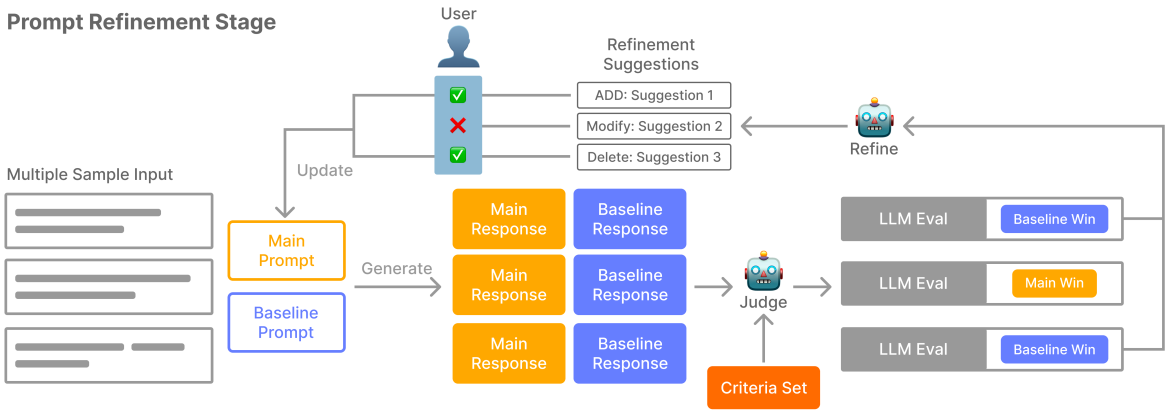

Figure 4: In the prompt refinement stage, EVALIGNER generates responses for multiple sample inputs. With the criteria set defined in the criteria refinement stage, a judge module evaluates the responses. If there are evaluations where the baseline wins, a refinement module suggests prompt refinements for the Main Prompt. The user can selectively apply the suggestions and update the Main Prompt.

If alignment is not achieved after three iterations, the criteria set from the third trial is presented to the user.

2.2.2 Prompt Refinement Pipeline (Fig. 4). To assist users to edit their prompt to perform as they intend, EVALIGNER provides an automatic prompt refinement pipeline. The system modifies the user's prompt at the sentence level, enabling iterative refinement and continuous testing. The refinement module receives the user's prompt, broken down at the sentence level. While the prompt could be decomposed using rule-based methods (e.g., splitting by periods), we use an LLM-based approach to deal with various text formats (e.g., JSON, Markdown) and unpredictable formats from user input. After decomposition is complete, the pipeline provides the refinement module with all the prompt sentences and LLM evaluation results. The refinement module applies only one of three actions—Addition, Modification, or Deletion—to a single sentence. By refining at the sentence level, the refinement module focuses on making discrete significant changes, instead of multiple less relevant changes. Therefore, the system suggests only the most necessary refinements.

2.3 Implementation

EVALIGNER is implemented with the frontend frameworks ReactJS and TypeScript. For all LLM components, we use the OpenAI API and `gpt 4o-2024-08-06`. We set the temperature to 0.3 for evaluation, 0.1 for the decomposition of the prompt, and 0.5 for all refinement requests. Through several pilot tests, we found that prompt decomposition performs better with a lower temperature, while refinement benefits from a higher temperature to generate diverse suggestions. All prompts use the *Chain-of-Thoughts* method [27], and, for prompt for decomposition, few-shot examples were given.

3 Future Work

In this workshop paper, we introduce the EVALIGNER system, which automatically refines the user's criteria set based on user feedback and improves users' prompts to better align with the user's criteria. This project is still in progress, and we have not yet conducted user studies. We are planning to conduct user studies to evaluate the design decisions behind EVALIGNER and assess whether the system effectively helps users align AI responses with their intended goals.

3.1 Appropriate Granularity Level for User Evaluation

When EVALIGNER makes users evaluate responses and give feedback, it is important to determine how detailed their feedback should be. There is a trade-off in deciding the granularity level of user evaluation in the evaluation process. If users provide feedback that is too coarse-grained (i.e., simply selecting the better one between two responses), LLMs may struggle to accurately infer the user's intention, potentially leading to misalignment over successive refinements. On the other hand, requiring users to give highly fine-grained feedback (i.e., evaluating responses for every sub-attribute in a criterion) increases users' workload, which may lead to fatigue. To balance this, EVALIGNER prompts users to provide unstructured natural language feedback to clarify misalignment, instead of evaluating every sub-attribute individually, and only prompts users when the LLM evaluation does not align with their evaluation. In future user studies, we aim to evaluate whether our granularity level design effectively reduces users' workload while providing sufficient guidance to align future LLM evaluations.

3.2 Support Reviewing Multiple Responses and Evaluations

In the criteria refinement stage, users focus on evaluating a single sample input. However, in the prompt refinement stage, responses are generated and evaluated across multiple sample inputs, making it difficult for users to manually check every response and evaluation. Several prior work explored ways to support users to

understand a large number of text through techniques such as highlighting, clustering, and visualization [8, 25]. In EVALIGNER, we assume that criteria have already been revised sufficiently where the user does not have to validate them in prompt refinement. However, the criteria still might have flaws that users should or want to check during prompt refinement. Through user studies, we plan to understand users' need to validate larger samples during the prompt refinement stage and, if there is such a need, what aspects users want to focus on. Based on this, we aim to explore techniques to support users in understanding and effectively evaluating a large number of LLM-generated responses.

3.3 Scope of Prompt Refinement

The current version of EVALIGNER refines prompts at the sentence level. This design is based on common practices in prompt engineering, where engineers iteratively make small changes in a prompt and test the response from the new prompt [33]. Based on this design rationale, we expect that sentence-level refinement will help users better understand refinement suggestions by making changes more manageable and easy to follow. However, prior work has explored various prompt engineering techniques (e.g., ReAct [29], few-shot prompting [4]), which suggest that modifying the overall structure of a prompt could also be valuable for prompt improvement. Through user studies, we will assess how sentence-level refinements in the prompt refinement stage help users iterate and improve on prompts. Beyond the current version of EVALIGNER, we also plan to explore an approach that combines sentence-level refinements with structural-level modifications (e.g., restructuring prompts to apply explicit reasoning steps). Through future user studies, we aim to evaluate which approach effectively helps users to better align the prompts with their goals.

References

[1] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L Glassman. 2024. ChainForge: A Visual Toolkit for Prompt Engineering and LLM Hypothesis Testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–18.

[2] Zahra Ashktorab, Michael Desmond, Qian Pan, James M Johnson, Martin Santillan Cooper, Elizabeth M Daly, Rahul Nair, Tejaswini Pedapati, Swapnaja Achintalwar, and Werner Geyer. 2024. Aligning Human and LLM Judgments: Insights from EvalAssist on Task-Specific Evaluations and AI-assisted Assessment Strategy Preferences. *arXiv preprint arXiv:2410.00873* (2024).

[3] Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2024. Benchmarking foundation models with language-model-as-an-examiner. *Advances in Neural Information Processing Systems* 36 (2024).

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating Summarization Evaluation. arXiv:2007.12626 [cs.CL] https://arxiv.org/abs/2007.12626

[6] Krzysztof Z Gajos and Lena Mamykina. 2022. Do people engage cognitively with AI? Impact of AI assistance on incidental learning. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*. 794–806.

[7] 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.

[8] Katy Ilonka Gero, Chelse Swoopes, Ziwei Gu, Jonathan K Kummerfeld, and Elena L Glassman. 2024. Supporting Sensemaking of Large Language Model Outputs at Scale. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–21.

[9] Daphne Ippolito, Ann Yuan, Andy Coenen, and Sehmon Burnam. 2022. Creative writing with an ai-powered writing assistant: Perspectives from professional writers. *arXiv preprint arXiv:2211.05030* (2022).

[10] Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024. Prometheus: Inducing Fine-grained Evaluation Capability in Language Models. arXiv:2310.08491 [cs.CL] https://arxiv.org/abs/2310.08491

[11] Tae Soo Kim, Yoonjoo Lee, Jamin Shin, Young-Ho Kim, and Juho Kim. 2024. Evallm: Interactive evaluation of large language model prompts on user-defined criteria. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–21.

[12] 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:2301.13298 [cs.CL] https://arxiv.org/abs/2301.13298

[13] Bishal Lamichhane. 2023. Evaluation of chatgpt for nlp-based mental health applications. *arXiv preprint arXiv:2303.15727* (2023).

[14] 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).

[15] Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. 2024. WILDBENCH: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild. *arXiv preprint arXiv:2406.04770* (2024).

[16] 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).

[17] 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. *Comput. Surveys* 55, 9 (2023), 1–35.

[18] Yuanye Liu, Jiahang Xu, Li Lyna Zhang, Qi Chen, Xuan Feng, Yang Chen, Zhongxin Guo, Yuqing Yang, and Peng Cheng. 2025. Beyond Prompt Content: Enhancing LLM Performance via Content-Format Integrated Prompt Optimization. arXiv:2502.04295 [cs.CL] https://arxiv.org/abs/2502.04295

[19] Yingzhou Lu, Minjie Shen, Huazheng Wang, Xiao Wang, Capucine van Rechem, Tianfan Fu, and Wenqi Wei. 2023. Machine learning for synthetic data generation: a review. *arXiv preprint arXiv:2302.04062* (2023).

[20] Vishakh Padmakumar and He He. 2023. Does Writing with Language Models Reduce Content Diversity? *arXiv preprint arXiv:2309.05196* (2023).

[21] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. arXiv:2305.03495 [cs.CL] https://arxiv.org/abs/2305.03495

[22] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927* (2024).

[23] Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. 1–14.

[24] Nikhil Sharma, Q Vera Liao, and Ziang Xiao. 2024. Generative Echo Chamber? Effect of LLM-Powered Search Systems on Diverse Information Seeking. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–17.

[25] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. 2024. Luminate: Structured Generation and Exploration of Design Space with Large Language Models for Human-AI Co-Creation. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–26.

[26] Ming Wang, Yuanzhong Liu, Xiaoyu Liang, Songlian Li, Yijie Huang, Xiaoming Zhang, Sijia Shen, Chaofeng Guan, Daling Wang, Shi Feng, Huaiwen Zhang, Yifei Zhang, Minghui Zheng, and Chi Zhang. 2024. LangGPT: Rethinking Structured Reusable Prompt Design Framework for LLMs from the Programming Language. arXiv:2402.16929 [cs.SE] https://arxiv.org/abs/2402.16929

[27] 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.

[28] Ryen W White. 2023. Navigating complex search tasks with AI copilots. *arXiv preprint arXiv:2311.01235* (2023).

[29] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022).

[30] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. 2023. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661* (2023).

[31] Seonghyeon Ye, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, Seungone Kim, Yongrae Jo, James Thorne, Juho Kim, and Minjoon Seo. 2024. FLASK: Fine-grained Language Model Evaluation based on Alignment Skill Sets. arXiv:2307.10928 [cs.CL] https://arxiv.org/abs/2307.10928

[32] JD 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*. 2206–2220.

[33] JD 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*. 1–21.

[34] Kechi Zhang, Jia Li, Ge Li, Xianjie Shi, and Zhi Jin. 2024. Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339* (2024).

[35] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).

[36] 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:2210.07197 [cs.CL] https://arxiv.org/abs/2210.07197