PrEAM: Prompt Optimization using Evaluations by Automated Micro-judges

Anonymous ACL submission

Abstract

Prompt quality plays an important role in the performance of LLM-powered QA (Question-Answering) systems. However, maintaining high-quality prompts remains a labor-intensive and fickle task. We introduce **PrEAM**, the first *continual* prompt optimization framework for QA tasks that makes use of automated LLM-as-judge feedback. PrEAM closes the loop between generation, evaluation and prompt improvement by processing and leveraging feedback from one or more specialized, LLM-based *micro-judges* that independently score every answer on each turn. For example, task-critical axes might include faithfulness, relevance, completeness, conciseness, among others. Within each micro-judge, errors are investigated and classified based on root cause. The top errors for each micro-judge are then aggregated into targeted edits of the system prompt. This process is repeated until the performance of the train and test set diverges, producing a self-healing system prompt that adapts as the knowledge base, user mix, or model version evolves. Using GPT 4o and GPT 4.1 on a dataset of 400 multi-turn QA tasks and an Arena-Hard dataset, respectively, shows marked improvement in just a handful of iterations while only requiring a few minutes to run.

1 Introduction

Large language models (LLMs) have significantly advanced the field of conversational question answering (QA), especially when paired with retrieval-augmented generation (RAG) pipelines that ground answers in an external knowledge store (Lewis et al., 2020).

LLMs have been found to be incredibly versatile, and through prompting, one can leverage an LLM's ability to predict and reason to accomplish a wide set of tasks. However, practitioners have quickly discovered a stubborn bottleneck: *prompt engineering*. Tiny changes in phrasing can decide whether a model cites evidence or hallucinates, whether it tracks dialogue history or forgets the previous turn (Bach et al., 2022). Because real-world knowledge bases and user intents drift constantly, even a well-crafted prompt degrades within weeks, forcing costly manual re-tuning.

As a solution, automatic prompt optimization methods have begun to appear. Gradient-guided token search (Shin et al., 2020), reinforcement learning approaches (Deng et al., 2022), and edit-based strategies (Prasad et al., 2023; Guo et al., 2023) all show promise, but either assume a stable reward signal or ignore feedback produced by powerful LLM evaluators.

An *LLM-as-a-judge* is a special case of deploying LLMs as automated graders. Recent work demonstrates great agreement with expert annotators in summarization, translation, and open-ended generation (Liu et al., 2023; Manakul et al., 2023; Gu et al., 2024; Zheng et al., 2023). However, how to best integrate such judges into a closed-loop prompt-engineering workflow remains underexplored. This line of evaluation has grown in popularity over the last year as it presents a way to conduct human-aligned automatic evaluation on an unprecedented scale. A common evaluation scheme for LLM-powered QA systems goes as follows: Given a set of user queries, the system is prompted using some predefined prompt to generate a set of responses to the queries. From there, those responses are passed through some LLM-as-a-judge to evaluate the responses in some fashion. This might include specific judges prompted to evaluate traditional metrics, such as query relevance or conciseness. This judge could potentially evaluate responses with respect to a set of ground truth responses created by humans or another LLM. When such judges are created, they are often instructed to provide an explanation for their decision in addition to their verdicts. This is a valuable signal for prompt improvement, but is often ignored in existing automatic prompt improvement approaches.

In this paper, we present **PrEAM**: Prompt optimization using Evaluations by Automated Micro-judges—a closed-loop framework that

continuously aligns the prompts for Question-Answering tasks by processing, collating, and leveraging feedback across one or more LLM-as-judges. PrEAM orchestrates a multistage pipeline that (i) decomposes evaluation into targeted dimensions (faithfulness, answerability, style, etc.), (ii) aggregates judge feedback into structured error categories, and (iii) synthesizes minimal, auditable prompt edits that directly target the dominant failure modes. This process repeats until prompt performance reaches a plateau/decreases or until prompt performance on a held-out test dataset starts to diverge from the performance on the dataset it is using to generate prompt improvement suggestions. The latter condition ensures that PrEAM does not overfit to the specific dataset on which it is trained. The system therefore learns from its own mistakes, with no gradients, no gold labels (unless required by the suite of judges), and no humans in the loop.

Contributions.

1. We propose a novel framework for prompt improvement that leverages the valuable signal produced by LLMs-as-judges. PrEAM does not require access to the model beyond black-box API calls and is fully human-out-of-the-loop.

2. The prompt improvement loop is fully auditable by humans, from the the judge feedback to the error categorization to the targeted areas of improvement. All aspects of PrEAM can be examined, not only for debugging purposes but also for use as an error summary by human prompt engineers.

3. Experiments conducted on an internal dataset of multi-turn RAG (Retrieval Augmented Generation) QA as well as on the widely-known Arena Hard dataset show drastic prompt improvements when using PrEAM, either matching or exceeding human performance in only a few iterations, taking a fraction of the time.

By demonstrating that an ensemble of LLMs can act as both *critic* and *coach*, PrEAM advances the vision of self-improving language agents that keep themselves aligned as their environment changes.

2 Related Work

2.1 Automatic Prompt Optimization

Early work explored gradient-based token search (AUTOPROMPT; Shin et al., 2020). More recent approaches employ gradient-free edits (Prasad et al., 2023), evolutionary strategies (Guo et al., 2023), or reinforcement learning (Deng et al., 2022). While effective for static classification or single-shot generation, these methods assume a stationary reward and struggle with multi-turn dialogue.

Contemporaneous systems such as PROMPTWIZARD (Agarwal et al., 2024) and CRISPO (He et al., 2024) add iterative critique-and-rewrite cycles, yet they employ a *single* general-purpose critic and evaluate mostly on summarization. In contrast, PrEAM targets retrieval-grounded *conversational* QA and leverages a panel of interpretable micro-judges, each specialized for a distinct failure mode, yielding more stable multi-objective optimization.

DSPy (Khattab et al., 2024), a popular declarative framework for building AI systems, offers tools for prompt optimization. Its techniques often rely on labeled examples, either authored by humans or generated by another language model. This enables optimization across an arbitrary graph of LM interactions. In contrast, PrEAM targets prompt refinement through explicit reasoning traces produced by micro judges. Its iterative, evolutionary design supports rapid adaptation to shifting user intents and evolving optimization metrics. The two approaches complement each other: DSPy supports end-to-end pipeline synthesis, while PrEAM handles continual prompt maintenance within production chat systems.

2.2 LLM-Based Evaluation and Self-Refinement

Large models such as GPT-4o (OpenAI, 2024b) have proven to be surprisingly reliable as automatic evaluators, closely correlated with expert judgments in summarization and translation (Liu et al., 2023). Self-refinement frameworks let a generator critique and revise its own output (Madaan et al., 2023; Shinn et al., 2023), while recent work ensembles multiple "LLM judges" to reduce evaluation variance (Rahmani et al., 2024). PrEAM extends this line of work by *embedding* LLM evaluators *within* the optimization loop for prompt design, not just for evaluation purposes.

2.3 Conversational QA with Retrieval

RAG pipelines are now standard for keeping QA responses grounded (Lewis et al., 2020; Shuster et al., 2021). Instruction tuning and few-shot prompting improve answer style (Ouyang et al., 2022), but

neither adapts automatically when the underlying corpus changes. Our work is the first to introduce a fully automated feedback loop where LLMs act as both judges and prompt engineers, continuously improving prompts in conversational QA without human intervention.

2.4 Summary

PrEAM uniquely combines (i) multi-aspect LLM evaluation with (ii) a continuous closed-loop prompt editing scheme leveraging LLM-as-a-judge reasoning traces. This integration has not, to the best of our knowledge, been explored in the prior art.

3 System Overview

In this section we give a concise end–to–end view of **PrEAM** and describe how its components interact in a closed feedback loop that continually improves a system prompt without human supervision. Figure 1 presents a high–level block diagram of the entire pipeline, while Figure 2 zooms in on the error–processing stages that transform raw judge outputs into actionable edits.

At the heart of PrEAM is the idea of using LLMs both as *evaluators* and *editors*. Specifically, we leverage *micro-judges*, LLM-based agents that independently evaluate dimensions of an answer at a particular turn of conversation, such as groundedness, relevance, completeness, or performance against some benchmark response or following a set of rules for an ideal response, and provide justifications in natural language along with categorical labels. Such evaluation schemes are common in practice, and are used frequently for their ability to scale arbitrarily while still being aligned with human preferences. These judgments are synthesized and passed to a separate LLM-based *meta-prompting module*, which proposes improvements to the system prompt in natural language. This idea builds on recent advances in meta-prompting and self-improving LLMs (Zhou et al., 2023; Prasad et al., 2023; Madaan et al., 2023), where models are guided to revise instructions based on performance feedback.

PrEAM consists of the following components:

- **Conversational QA Module:** A Question-Answering pipeline built on a pre-trained LLM (e.g., GPT-4o), which generates responses to user queries using a given input prompt. For example, one might have a Conversational RAG (Retrieval Augmented Generation) pipeline that uses retrieved context and conversation history to answer a user's questions about a specific Knowledge Base. Alternatively, it could simply be a QA system in which the model answers user queries directly without any additional content.

- **LLM-Based Micro-Judges:** A set of specialized LLM-based evaluators, each tasked with assessing model responses along a distinct dimension—such as *groundedness*, *relevance*, or *formatting*. For each dimension, the corresponding micro-judge produces both a discrete verdict (*Acceptable*/*Unacceptable*) and a natural language rationale explaining its judgment. This auto-evaluation process across multiple micro-judges for a single response is illustrated in Fig. 3.

- **Error Summarizer:** Aggregates the rationales from all failed or unacceptable cases into short, instance–agnostic summaries as shown in Fig. 2.

- **Error Categoriser:** Group summaries into a small set of categories of recurring errors (Fig. 2).

- **Prompt Optimizer (Meta-Prompting Module):** An LLM instance receives the original prompt, common failure categories and representative examples and outputs an improved version of the prompt.

- **Optimization Loop Controller:** A control mechanism that orchestrates each iteration: generating answers with the latest prompt, re-evaluating them via micro-judges, summarizing feedback, and applying meta-prompting while performance on a held-out test set improves. In our setup, we split the responses into an 80-20 train-test split. We use 80% train split to optimize the prompt and reserve the 20% test split to evaluate performance on a given prompt.

This architecture allows for prompt refinement without any human-in-the-loop supervision. Unlike one-shot tuning approaches, PrEAM can continuously adapt the prompts to new domains or changing user behaviors. By combining fine-grained

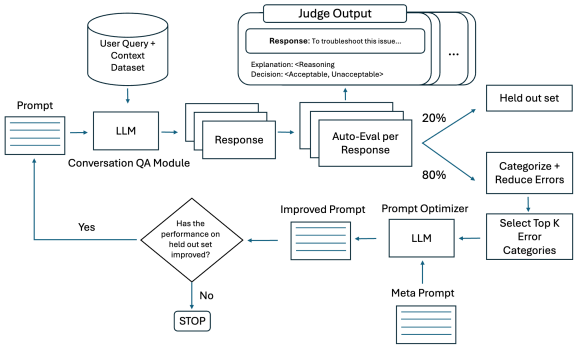

Figure 1: PrEAM Pipeline overview.

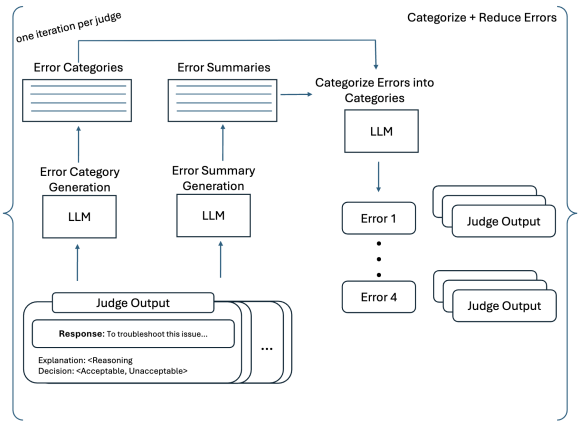

Figure 2: Error categorization and error reduction.

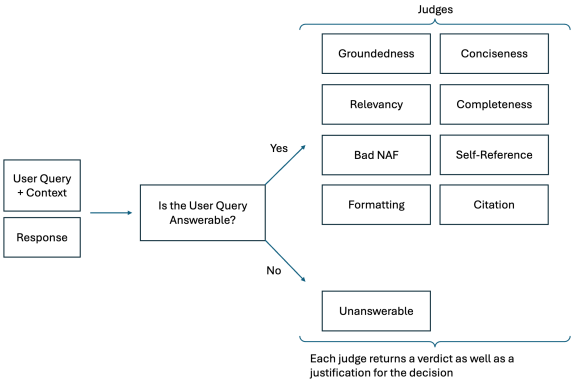

Figure 3: Single Response Auto-Evaluation by a Judge Model

LLM-based evaluation with automated prompt revision, PrEAM offers a scalable general-purpose solution to improve instruction quality in QA systems. In the next section, we will go over more details on the components of PrEAM discussed above.

4 Feedback-Driven Optimization Loop

The core innovation of PrEAM lies in its iterative prompt refinement mechanism. This section details how prompts are evaluated and improved through automated LLM-based feedback.

4.1 Initial Prompt Definition

The iterative process starts with an initial baseline prompt, typically a manually crafted system prompt for the QA system. This can be an extremely simple one-sentence description of the task. For example: *"Answer the following question:"* This prompt is fixed for the first iteration. Subsequently, the system is executed on a training subset of conversational data, which may also encompass context for each query, facilitating the generation of responses for each user interaction.

4.2 Evaluation by Micro-Judges

Each answer is evaluated using one or more LLM-based micro-judges. Note that this evaluation occurs over a single user turn, hence the name *micro-judge*. In many enterprise use-cases, the same conversational QA prompt is used multiple times in succession as a conversation flows. PrEAM focuses on evaluations that center on improving the performance of individual turns in this multi-turn setting, not the conversation as a whole. However, applying judge evaluation and prompt improvement on a conversation trace is a future area of development for PrEAM.

Each judge provides a brief explanation of its reasoning and returns a score or decision. For cases in which the decision is deemed a failure, the explanation is used in future steps. Note that the existence of such an explanation before the verdict is not an abnormal requirement, as most LLM-as-judges use CoT (Chain of Thought) prompting (Wei et al., 2022), which encourages models to give a coherent thought process before coming to conclusions. For each sample, the explanation and verdict are logged.

4.3 Categorization of Failure Cases

All low-scoring responses (e.g. below a threshold on any metric or when "decision" is unacceptable) undergo three steps: Error Summarization, Error Category Generation, and Error Categorization.

Firstly, the judge feedback for each error case is summarized to remove test-case specific details. If the raw judge output does not function well as direct feedback (for example, judges that function

by comparing the model response against another model's response), this step will pre-process the thoughts in a way that is digestable by future steps in the PrEAM pipeline.

PrEAM then contains an error category generation prompt that ingests the summarized feedback for all error cases and outputs a set of 3-5 error categories per micro judge. For example, a judge measuring the preference of a model output with respect to a baseline response could have multiple reasons for passing a failing verdict. The response could be too long, contain incorrect information, or have an inappropriate voice or tone. Note that even in cases where there are multiple micro judges, it is possible for each micro judge to have multiple error categories.

Then each error is categorized into the category that most accurately explains it. Once all errors have been mapped to a category, all categories across all judges are then compiled to determine the most common reasons for error. The categories with the Top-k errors are then forwarded to the meta-prompt, which uses this information to generate a better prompt.

4.4 Meta-Prompted Prompt Optimization

The categorized feedback is passed to a meta-prompting module, another LLM instance tasked with prompt rewriting. This module receives:

- The original prompt.
- A list of the Top-k failure categories. Each failure category description contains:
  - The name of the error category (ex. "Model Response not Grounded in Provided Context")
  - A short description of the error category (ex. "The model response contains incorrect information not present in the provided context.")

Then it generates a revised prompt tailored to mitigate the observed issues. For example, if many hallucinations occurred, the revised prompt might emphasize stronger adherence to source content: *"If the answer is not explicitly stated in the sources, say you do not know."*

4.5 Iteration and Convergence

The newly generated prompt is then used to rerun the system on the training data. The micro-judges re-evaluate the updated responses. This process is repeated for several iterations (typically 2–3), allowing the system to progressively improve. Convergence is declared when either:

- Metric scores plateau/decrease across iterations, or
- Performance on the held-out validation set begins to diverge from training gains, indicating overfitting.

At convergence, the best-performing prompt is selected for final evaluation on the test set.

5 Experimental Setup

We evaluate PrEAM on two question-answering datasets: an internal enterprise domain multi-turn retrieval-augmented QA dataset, and the public **Arena-Hard-v2.0-Preview** benchmark (Li et al., 2024; Tianle Li*, 2024). Below, we provide details on the datasets, evaluation metrics, and experimental protocols.

Internal RAG QA Dataset. This dataset consists of 400 multi-turn dialogues. For each user query, relevant context passages are pre-retrieved from a knowledge base, and the LLM must generate an answer using both the conversation history and the retrieved context. We evaluate each response across several task-critical axes commonly used in retrieval-augmented generation: *Groundedness*, *Relevance*, *Formatting*, *Conciseness*, *Completeness*, *Citation*, and *Self-Referentiality*. Each criterion is assessed by a dedicated LLM-based *micro-judge*, with the judging prompts calibrated on a human-annotated preference dataset. For answerable queries, each judge returns a rating of *unacceptable*, *acceptable*, or *ideal*. For unanswerable queries, we include an additional check to ensure that the answer explicitly acknowledges that it cannot answer the question (for example, an apologetic statement in the first sentence). A response is considered *failure case* on any given axis if it is rated *unacceptable* by the corresponding judge. We define an overall **Aggregate Decision** metric as the percentage of responses that pass *all* judges (i.e., no axis is marked unacceptable).

We run PrEAM's prompt-refinement loop on this internal dataset, using a portion of the 400 dialogues for iterative prompt tuning and holding out the rest for evaluation. Prompt updates continue until the Aggregate Decision on the held-out

test set plateaus or declines (indicating that performance has peaked and further tuning would risk overfitting). In practice, this stopping criterion was triggered after three refinement iterations, resulting in a final optimized prompt at iteration 3.

Arena-Hard Benchmark. To evaluate generalization beyond our internal data, we apply PrEAM to the **Arena-Hard-v2.0-Preview** benchmark. Arena-Hard comprises 750 challenging open-domain English questions (500 focused on coding/math problems and 250 on creative writing tasks) sourced from the Chatbot Arena platform. Each question is paired with a strong baseline answer for comparison. We employ GPT-4.1 (OpenAI, 2024a) as an automated judge in a pairwise evaluation setting: given the model's response and the baseline answer, the judge assigns a comparative verdict—*much better*, *better*, *about the same*, *worse*, or *much worse*—for the model's answer relative to the baseline. We convert these judgments into a **weighted win rate** for our model, where a "much better" verdict contributes more strongly than a "better" verdict, and any "worse" or "much worse" verdict counts as a loss. In the context of PrEAM, any instance where the model's answer is judged *worse* or *much worse* than the baseline is treated as a failure case. Along with the verdict, the LLM judge also provides a natural-language rationale by first synthesizing its own ideal answer and then analyzing the differences between the model's answer and the baseline. We distill this comparative feedback into direct instructions for the next prompt revision. As with the internal data, we iterate prompt optimization on a subset of Arena-Hard tasks until the weighted win rate on a held-out set no longer improves.

6 Results

We report PrEAM's performance on the internal RAG QA dataset and the Arena-Hard benchmark. In both settings, iterative prompt optimization yields substantial gains over the initial prompt, often in just a few refinement rounds.

6.1 Internal RAG QA Dataset

Tables 1 and 2 summarize the performance of the original prompt and successive refined prompts on the internal dataset's training and test splits, respectively. The original prompt produces low overall quality, with especially poor results on axes like *Citation* (only 2.01% of training responses and 0.00% of test responses meet the citation requirement) and *Unanswerability* (29.01% train, 17.54% test, indicating the system often fails to acknowledge when it cannot answer). After a single PrEAM iteration, we observe substantial improvements on most axes: for instance, the Citation metric jumps to **79.86%** on train and **70.77%** on test, and Unanswerability rises to **93.13%** on train and **77.19%** on test. By the third iteration, many metrics reach their peak and we notice a more balanced performance—e.g., Citation improves to **95.30%** (train) and **92.31%** (test) without dropping the other metrics significantly—reflecting a dramatic enhancement in answer quality compared to the original prompt. Note that our goal is to have a balanced performance across multiple axes, and we will illustrate this in the next paragraph.

Not every metric improves monotonically; some dimensions trade off against others. For example, making answers more *concise* can conflict with *completeness*, since an extremely brief answer may omit details necessary for completeness. Indeed, we observe that certain metrics (such as *Completeness*) dip slightly in later iterations even as others (e.g., *Conciseness*) continue to improve. For example, in the case of Completeness and Conciseness, this tradeoff can be explained by the fact that more thorough responses tend to be more complete, but less concise. Given these trade-offs, we rely on the *Aggregate Decision* as an overall indicator of success. The Aggregate Decision—i.e., the proportion of responses rated acceptable or ideal on *all* criteria—rises from only **14.64%** with the original prompt to **83.21%** after three iterations on the training set. On the held-out test set, it climbs from **8.2%** to **76.23%** by iteration 3 (Table 2). After the third iteration, further prompt edits yielded no consistent gains (and even led to slight degradations on many axes), so we selected the iteration-3 prompt as the final optimized prompt.

Please refer to Appendix 9 for an example iteration of a prompt improvement. While the original prompt contains a simple instruction and relies on the model's inherent quality to produce an acceptable output, the prompt after the first iteration of prompt improvement explicitly touches on evaluation areas of the micro-judges and guides the model into producing better outputs. It also contains sections on output format and a section for final reminders, both well-established practices in the field of prompt engineering.

Metric	Orig.	Iter 1	Iter 2	Iter 3	Iter 4
Citation	2.01	79.86	89.93	**95.30**	91.28
Completeness	**96.64**	75.17	88.59	85.91	76.51
Conciseness	85.91	**100.00**	97.32	95.97	95.30
Formatting	**100.00**	96.64	95.97	99.33	97.32
Groundedness	**99.33**	93.96	97.31	97.32	91.27
Relevance	**100.00**	92.62	97.31	97.31	85.23
Self-Reference	**92.19**	82.35	95.45	95.45	83.78
Unanswerability	29.01	93.13	77.86	85.50	**91.60**
Aggregate	14.64	77.50	76.79	**83.21**	77.86

Table 1: Internal dataset (train split) performance across prompt refinement iterations. Best results per metric are bolded and the highlighted column represents the final prompt after stopping criteria has been reached.

Metric	Orig.	Iter 1	Iter 2	Iter 3	Iter 4
Citation	0.00	70.77	87.69	**92.31**	86.15
Completeness	**92.31**	76.92	86.15	81.54	63.08
Conciseness	89.23	**100.00**	98.46	96.92	96.92
Formatting	**98.46**	96.92	98.46	98.46	96.92
Groundedness	**96.92**	84.62	96.92	96.93	90.77
Relevance	**98.46**	80.00	96.93	95.38	87.69
Self-Reference	**100.00**	63.64	86.67	76.92	85.71
Unanswerability	17.54	77.19	70.18	80.70	**92.98**
Aggregate	8.20	65.57	71.31	**76.23**	72.95

Table 2: Internal dataset (test split) performance across prompt refinement iterations. Best results per metric are bolded and the highlighted column represents the final prompt after stopping criteria has been reached.

6.2 Arena-Hard Evaluation

We further test PrEAM's generalizability using **Arena-Hard-v2.0-Preview**, where the evaluation is comparative: model outputs are judged relative to a strong baseline using GPT-4.1 as a referee. Table 3 shows PrEAM's performance progression on the Arena-Hard benchmark. With the original prompt, our model underperforms the baseline, achieving a weighted win rate of only **32.4%** (indicating that it loses to the baseline in the majority of comparisons). After two rounds of prompt refinement, the model's score improves to **51.7%**, now surpassing the baseline. The refined prompt is especially effective on the coding and math problems in Arena-Hard, where precise, well-grounded answers are critical, yielding large gains in those categories. Improvements on the creative writing tasks are more modest, likely because these tasks are highly subjective and the baseline answers do not incur heavy penalties for creative elaboration (making it harder to strongly outperform them). Nonetheless, these results demonstrate that PrEAM's benefits carry over to challenging unseen tasks. Without any gradient updates or human-in-the-loop intervention, our prompt-only optimization approach manages to exceed a strong baseline model on a widely challenging benchmark.

Split	Orig.	Iter 1	Iter 2	Iter 3
Train	0.52	0.76	**0.77**	0.72
Test	0.39	0.52	**0.62**	0.56

Table 3: Arena-Hard weighted win rates across prompt iterations. Best scores per row are bolded and the highlighted column represents the final prompt after stopping criteria has been reached.

7 Conclusion

We introduced PrEAM, a novel framework for prompt optimization in conversational question answering systems that leverages evaluations by automated micro-judges. By integrating LLM-based feedback into a closed-loop optimization process, PrEAM enables self-improving prompt design without human intervention. Each component of the system from judgment to revision is powered by LLMs, creating a scalable mechanism for dynamic prompt adaptation in real-world, retrieval-augmented QA scenarios.

In our internal data set, the aggregate performance improved from 14.64% in the original prompt to 83.21% after three iterations in the training split and from 8.20% to 76.23% in the test split. These gains were reflected across key metrics such as citation accuracy, completeness, relevance, etc. Furthermore, the evaluations on the Arena-Hard benchmark revealed that our refined prompts achieved a weighted win rate of 77% on the training split and 62% on the test split, outperforming the original prompt's 52% and 39%, respectively. These results underscore the generalizability and robustness of our prompt optimization approach.

Our findings highlight the viability of using LLMs not just as generators but also as evaluators and editors. By decomposing evaluation into interpretable sub-tasks and leveraging multiple specialized micro-judges, PrEAM avoids over-optimizing for a single metric and instead balances multiple quality dimensions. The result is a generalizable and domain-adaptable prompt optimization method that can enhance performance across diverse conversational settings.

Future work will explore broader applications of micro-judge-guided prompt optimization beyond QA, including summarization, dialog policy

generation, and safety-critical domains. In addition, incorporating human-in-the-loop oversight or user satisfaction signals could further enhance prompt quality and alignment. As LLMs continue to evolve, systems like PrEAM offer a path toward more robust, transparent, and self-improving AI pipelines.

8 Limitations

Although PrEAM removes the human from the critical path, it still depends on the quality of the upstream judges. Misaligned evaluators can steer optimization astray and effectiveness can vary between domains.

9 Ethical Considerations

Automating prompt engineering raises questions about accountability and bias. We recommend periodic human audits and explicit fairness checks on judge prompts.

References

Eshaan Agarwal, Joykirat Singh, Vivek Dani, Raghav Magazine, Tanuja Ganu, and Akshay Nambi. 2024. Promptwizard: Task-aware prompt optimization framework. *arXiv preprint arXiv:2405.18369*.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, et al. 2022. Promptsource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, et al. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3369–3391.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, et al. 2024. Llm-as-a-judge: A comprehensive survey on llm-based evaluators. *arXiv preprint arXiv:2411.15594*.

Qingyan Guo et al. 2023. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*.

Han He, Qianchu Liu, Lei Xu, Chaitanya Shivade, Yi Zhang, Sundararajan Srinivasan, and Katrin Kirchhoff. 2024. Crispo: Multi-aspect critique–suggestion-guided automatic prompt optimization for text generation. *arXiv preprint arXiv:2410.02748*.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. Dspy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Vladimir Karpukhin, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2511–2522.

Aman Madaan et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

OpenAI. 2024a. Gpt-4.1 technical overview. https://openai.com. Accessed: 2025-05-18.

OpenAI. 2024b. Gpt-4o technical report. https://openai.com/index/gpt-4o. Accessed: 2025-05-16.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2023. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 3845–3864.

Hossein A. Rahmani, Emine Yilmaz, Nick Craswell, and Bhaskar Mitra. 2024. Judgeblender: Ensembling judgments for automatic relevance assessment. *arXiv preprint arXiv:2412.13268*.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Noah Shinn et al. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803.

Evan Frick Lisa Dunlap Banghua Zhu Joseph E. Gonzalez Ion Stoica Tianle Li*, Wei-Lin Chiang*. 2024. From live data to high-quality benchmarks: The arena-hard pipeline.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yi Zheng, Zeming Liu, Aohan Zeng, Canwen Xu, Zhiyuan Liu, Zhengyan Zhang, Zhen Wang, Biao Tang, Xiang Zeng, Yankai Li, et al. 2023. Judging llm-as-a-judge with mt-bench and arena. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Wendy Zhou, Timo Schick, Daniel Khashabi, and Dan Roth. 2023. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2307.10169*.

Appendix: Prompt Development History

Here we provide an example of prompt development using PrEAM:

Section	Content
Initial Prompt	`Answer the last user query in the conversation history.`
Error Summary Iteration 1	`- Missing Required Citations:` `The response does not include any citations in the required formats (such as [doc_1], (doc_1), [[cat_1]], etc.), which is mandatory for a valid answer. To correct this, the response must reference at least one knowledge base document or catalog item using the specified citation formats. Links, quoted content, or plain text references do not count as valid citations.`  `- Directly Answered or Attempted to Answer:` `The agent's response provided a direct, relevant, or partial answer to the user's question instead of stating that no answer or information is available. For a response to be considered 'ideal' unanswerable, it must begin by explicitly stating the absence of an answer or information. To correct this, the agent should not attempt to answer but should clearly state at the start that no answer is available.`  `- Lack of Explicit Unanswerability Statement at the Beginning:` `The agent's response did not begin with a clear statement indicating that no answer or information is available, as required by the 'ideal' conditions. Instead, the response may have provided general guidance, suggestions, or asked for clarification, but failed to directly acknowledge the lack of an answer at the start. To be 'ideal', the response must start with an explicit statement about the absence of an answer.`  `- Excessive Detail and Specificity:` `The response provides more information than necessary for a general or ambiguous query, including step-by-step instructions, specific dates, eligibility criteria, technical specifications, or procedural steps. To improve, responses should offer a high-level overview or summary, avoiding unnecessary specifics unless the user's question clearly requires them.`  `- Incorrect Citation Format:` `The response attempts to reference documents or catalog items but does so using an incorrect format (such as hyperlinks, bold text, or plain text), rather than the required citation formats. To fix this, all references to documents or catalog items must use the specified formats like [doc_1], (cat_1), or [[doc_1]].`  `- Redundancy and Repetition:` `The response repeats information or includes redundant statements, making it unnecessarily lengthy and less concise. To improve, responses should avoid repeating the same information and focus on delivering each point only once in a clear and succinct manner.`

New Prompt After Iteration 1	Answer the last user query in the conversation history. When responding, ensure the following: 1. **Citations**: Every response must include at least one properly formatted citation referencing the knowledge base documents or catalog items. Use the formats [doc_1], (doc_1), or [[cat_1]] as appropriate. If no relevant information is available, explicitly state this and do not fabricate citations. 2. **Unanswerability**: If the query cannot be answered based on the provided documents or catalog items, begin your response by explicitly stating that no answer or information is available. Do not attempt to provide partial answers, suggestions, or ask for clarification in such cases. 3. **Conciseness**: Avoid excessive detail or unnecessary specificity unless explicitly requested by the user. Provide a high-level overview or summary for broad or unclear queries. 4. **Clarity**: Avoid redundancy and repetition. Deliver each piece of information only once to ensure the response is clear and concise. 5. **Citation Format**: Ensure all references to documents or catalog items strictly follow the required citation formats ([doc_1], (doc_1), [[cat_1]]). Replace any non-compliant references with the correct format. Output Format - Begin with a clear statement of unanswerability if applicable. - Include at least one properly formatted citation if relevant information is available. - Ensure the response is concise, clear, and free of redundancy. Notes - Do not fabricate information or citations. - If the query is ambiguous or broad, provide a general response without unnecessary specifics. - Always prioritize clarity and adherence to the specified citation formats.

Error Summary Iteration 2	- Failure to Reference Relevant Information: The response fails to utilize or mention relevant information, documents, or catalog items that are available and directly address the user's query. This includes cases where the agent incorrectly claims no information is available, omits key resources, or does not guide the user to helpful next steps. To improve, the agent should always check for and incorporate pertinent details from the provided context, including catalog items and documents, and offer actionable guidance. - Missing Citations: The response does not include any citations in the required formats (e.g., [doc_1], [cat_1], (doc_1), etc.). Every response must contain at least one properly formatted citation, even when stating that no information is available. To correct this, ensure that all responses reference the relevant documents or catalog items using the specified citation formats. - Overlooking Relevant Information: The response incorrectly claims that no relevant information is available, even though the provided documents or catalog items contain information that directly addresses the user's query. This error occurs when the agent fails to utilize or reference available resources, leading to an inaccurate or incomplete answer. To avoid this, the agent should thoroughly check all provided materials and ensure any relevant content is incorporated into the response. - Incomplete or Inaccurate Summarization: The response provides only a partial answer, omits important details, or inaccurately summarizes the available information, leading to an incomplete or misleading response. This includes missing key policy points, eligibility criteria, or relevant document content. To improve, the agent should ensure all critical aspects of the user's query are addressed by thoroughly summarizing and including all relevant details from the context. - Incorrect Citation Format: The response includes citations, but they are not in the allowed formats. Common errors include using double square brackets for catalog items (e.g., [[cat_1]]), using hyperlinks instead of citations, or any other format not specified in the guidelines. To fix this, ensure all citations use only the approved formats such as [doc_1], [cat_1], (doc_1), etc., and avoid double brackets or links. - Directly Answered User's Question: The agent's response provides a direct or detailed answer to the user's question, rather than indicating that no answer or information is available. For a response to be considered 'ideal' in an unanswerable scenario, it must begin by explicitly stating that no answer or information was found. The agent should avoid providing an answer when the correct behavior is to acknowledge the absence of information.

New Prompt After Iteration 2	Answer the last user query in the conversation history. When responding, ensure the following: 1. **Citations**: Every response must include at least one properly formatted citation referencing the knowledge base documents or catalog items. Use the formats [doc_1], (doc_1), or [[cat_1]] as appropriate. If no relevant information is available, explicitly state this and do not fabricate citations. 2. **Thorough Use of Context**: Carefully review all provided documents, catalog items, and conversation history to identify relevant information. Do not overlook any details that could assist in answering the query. 3. **Unanswerability**: If the query cannot be answered based on the provided documents or catalog items, begin your response by explicitly stating that no answer or information is available. Do not attempt to provide partial answers, suggestions, or ask for clarification in such cases. 4. **Conciseness**: Avoid excessive detail or unnecessary specificity unless explicitly requested by the user. Provide a high-level overview or summary for broad or unclear queries. 5. **Clarity**: Avoid redundancy and repetition. Deliver each piece of information only once to ensure the response is clear and concise. 6. **Citation Format**: Ensure all references to documents or catalog items strictly follow the required citation formats ([doc_1], (doc_1), or [[cat_1]]). Replace any non-compliant references with the correct format. 7. **Actionable Steps for Overlooked Information**: If relevant information is found in the provided context but not directly addressed in the response, ensure it is incorporated. Do not claim that no information is available if relevant details exist. Output Format - Begin with a clear statement of unanswerability if applicable. - Include at least one properly formatted citation if relevant information is available. - Ensure the response is concise, clear, and free of redundancy. Notes - Do not fabricate information or citations. - If the query is ambiguous or broad, provide a general response without unnecessary specifics. - Always prioritize clarity and adherence to the specified citation formats. - If relevant information is overlooked, review the context again and ensure it is included in the response.

Error Summary Iteration 3	- Response Provides an Answer Instead of Stating Lack of Information: The agent's response directly answers the user's question or provides relevant information, rather than beginning with a clear statement that no answer or information is available. For a response to be considered 'ideal' unanswerable, it must explicitly state at the start that no answer or relevant information is found. The behavior that needs to change is that the agent should not attempt to answer or provide guidance if the required information is not available, but should instead clearly state the lack of information at the beginning of the response. - Failure to Utilize Relevant Information: The response fails to reference or incorporate relevant details from the provided documents or catalog items that directly address the user's query. This includes overlooking specific instructions, procedures, or resources that are available and applicable. To improve, the agent should thoroughly review the context and ensure all pertinent information is included in the response. - Incorrect Citation Format: This category includes responses where citations are present but do not follow the required formats ([doc_1], [cat_1], (doc_1), etc.), such as using double square brackets ([[cat_1]]), or any other non-approved format. To correct this, all citations must strictly adhere to the specified formats. - Missing Required Citations: This category covers responses that do not include any citations in the required formats, even though citations are expected. The behavior that needs to change is to ensure that at least one properly formatted citation is present in every response where citations are required. - Response Fails to Begin with Explicit Statement of No Information: The agent's response may eventually mention the lack of information, but does not begin with a clear statement that there is no answer or relevant information available. For a response to be 'ideal' unanswerable, this disclaimer must be at the very start. The behavior that needs to change is that the agent must always start the response with a direct statement about the absence of information if the answer cannot be provided. - Lack of Specificity or Actionable Guidance: The response is too vague or generic, lacking clear, actionable steps or instructions that the user can follow. This includes failing to provide direct links, step-by-step processes, or explicit references to catalog items or forms that would help the user complete their task. The agent should ensure responses are concrete and guide the user on exactly what to do next.

New Prompt After Iteration 3	Answer the last user query in the conversation history. When responding, ensure the following: 1. **Citations**: Every response must include at least one properly formatted citation referencing the knowledge base documents or catalog items. Use the formats [doc_1], (doc_1), or [cat_1] as appropriate. If no relevant information is available, explicitly state this and do not fabricate citations. 2. **Explicit Unanswerability**: If the query cannot be answered based on the provided documents or catalog items, begin your response by explicitly stating that no answer or information is available. Do not attempt to provide partial answers, suggestions, or ask for clarification in such cases. 3. **Thorough Use of Context**: Carefully review all provided documents, catalog items, and conversation history to identify relevant information. Do not overlook any details that could assist in answering the query. If relevant information is found in the provided context but not directly addressed in the response, ensure it is incorporated. 4. **Conciseness**: Avoid excessive detail or unnecessary specificity unless explicitly requested by the user. Provide a high-level overview or summary for broad or unclear queries. 5. **Clarity**: Avoid redundancy and repetition. Deliver each piece of information only once to ensure the response is clear and concise. 6. **Citation Format**: Ensure all references to documents or catalog items strictly follow the required citation formats ([doc_1], (doc_1), or [cat_1]). Replace any non-compliant references with the correct format. 7. **Actionable Steps for Overlooked Information**: If relevant information is found in the provided context but not directly addressed in the response, ensure it is incorporated. Do not claim that no information is available if relevant details exist. Output Format - Begin with a clear statement of unanswerability if applicable. - Include at least one properly formatted citation if relevant information is available. - Ensure the response is concise, clear, and free of redundancy. Notes - Do not fabricate information or citations. - If the query is ambiguous or broad, provide a general response without unnecessary specifics. - Always prioritize clarity and adherence to the specified citation formats. - If relevant information is overlooked, review the context again and ensure it is included in the response.

Error Summary Iteration 4	- Directly Answered Instead of Marking Unanswerable: The agent's response provides a direct or specific answer to the user's question, rather than stating at the beginning that no answer or information is available. For an 'ideal' unanswerable response, the agent must clearly state at the start that no answer or relevant information was found in the provided documents. The behavior that needs to change is that the agent should not attempt to answer the question if the answer is not available, but instead immediately indicate the lack of information. - Failure to Reference Relevant Documents or Catalog Items: The response does not utilize or mention available documents or catalog items that contain information directly related to the user's query. This includes cases where the agent claims no information is available, overlooks key resources, or fails to guide the user to actionable next steps. To improve, the agent should always check for and reference any relevant documents or catalog items, providing direct guidance or links when possible. - Omission of Specific Details or Steps: The response is too vague or general, lacking specific instructions, steps, or details that are available in the provided context. This includes not providing step-by-step guidance, missing key program dates, or failing to include all relevant benefits or features. To improve, the agent should extract and present all pertinent details from the documents, ensuring the user receives a thorough and actionable answer. - Missing Citations: The response does not include any citations in the required formats ([doc_1], (doc_1), [[doc_1]], [cat_1], etc.), even when citations are necessary. To correct this, the response must include at least one properly formatted citation whenever referencing information from the knowledge base or catalog items. - Incorrect or Inaccurate Information: The response provides information that is incorrect or contradicts the available documents, such as stating that no information exists when it does, or misrepresenting the content of the documents. To improve, the agent should carefully verify the information in the documents before responding and avoid making unsupported claims. - Overly Detailed and Specific Responses: The response includes excessive details, such as step-by-step processes, specific scenarios, or unnecessary information, instead of providing a concise, high-level summary. For general or ambiguous queries, the answer should avoid diving into specifics and instead offer a brief overview or direct next steps. To improve, responses should match the level of detail to the question's specificity and avoid including information that may not be relevant to the user's intent.

New Prompt After Iteration 4	Answer the last user query in the conversation history. When responding, ensure the following: 1. **Citations**: Every response must include at least one properly formatted citation referencing the knowledge base documents or catalog items. Use the formats [doc_1], (doc_1), or [cat_1] as appropriate. If no relevant information is available, explicitly state this and do not fabricate citations. 2. **Explicit Unanswerability**: If the query cannot be answered based on the provided documents or catalog items, begin your response by explicitly stating that no answer or information is available. Do not attempt to provide partial answers, suggestions, or ask for clarification in such cases. 3. **Thorough Use of Context**: Carefully review all provided documents, catalog items, and conversation history to identify relevant information. Do not overlook any details that could assist in answering the query. If relevant information is found in the provided context but not directly addressed in the response, ensure it is incorporated. 4. **Conciseness**: Avoid excessive detail or unnecessary specificity unless explicitly requested by the user. Provide a high-level overview or summary for broad or unclear queries. 5. **Clarity**: Avoid redundancy and repetition. Deliver each piece of information only once to ensure the response is clear and concise. 6. **Citation Format**: Ensure all references to documents or catalog items strictly follow the required citation formats ([doc_1], (doc_1), or [cat_1]). Replace any non-compliant references with the correct format. 7. **Actionable Steps for Overlooked Information**: If relevant information is found in the provided context but not directly addressed in the response, ensure it is incorporated. Do not claim that no information is available if relevant details exist. 8. **Prioritize Unanswerability Disclaimer**: Always begin the response with a clear statement of unanswerability if no relevant information is available. Do not provide context, general advice, or other information before this disclaimer. Output Format - Begin with a clear statement of unanswerability if applicable. - Include at least one properly formatted citation if relevant information is available. - Ensure the response is concise, clear, and free of redundancy. Notes - Do not fabricate information or citations. - If the query is ambiguous or broad, provide a general response without unnecessary specifics. - Always prioritize clarity and adherence to the specified citation formats. - If relevant information is overlooked, review the context again and ensure it is included in the response. - Ensure that the unanswerability disclaimer, if applicable, is the first statement in the response.