CLEAR: Error Analysis via LLM-as-a-Judge Made Easy

Asaf Yehudai[*1,2], **Lilach Eden**[*1], **Yotam Perlitz**[1], **Roy Bar-Haim**[1], **Michal Shmueli-Scheuer**[1]

[1]IBM Research
[2]The Hebrew University of Jerusalem
Asaf.Yehudai@ibm.com, lilache@il.ibm.com, Y.Perlitz@ibm.com, roybar@il.ibm.com, shmueli@il.ibm.com

Abstract

The evaluation of Large Language Models (LLMs) increasingly relies on other LLMs acting as judges. However, current evaluation paradigms typically yield a single score or ranking, answering *which* model is better but not *why*. While essential for benchmarking, these top-level scores obscure the specific, actionable reasons behind a model's performance. To bridge this gap, we introduce CLEAR, an interactive, open-source package for LLM-based error analysis. CLEAR first generates per-instance textual feedback, then it creates a set of system-level error issues, and quantifies the prevalence of each identified issue. Our package also provides an interactive dashboard that supports a comprehensive error analysis. We demonstrate CLEAR analysis for RAG and Math benchmarks, and showcase its utility through a user case study.

Code — https://ibm.biz/CLEAR-code-repo

Introduction

Evaluation of generative AI increasingly adopts the LLM-as-a-Judge (LLMaJ) paradigm (Zheng et al. 2023), where LLMs automatically score or compare system outputs (Liu et al. 2023). Aggregated judgments enable benchmarking and ranking of models (Gera et al. 2025), but they provide little insight into *why* systems fail. Developers still rely on tedious manual inspection to identify recurring issues, understand the system's current limitations, and effectively prioritize improvements.

Existing error analysis approaches only partially address this gap. Some rely on labeled datasets or predefined taxonomies (Wu et al. 2019; Murahari et al. 2023), while others analyze only the input questions and do not use the model responses (Zeng et al. 2025). These methods limit scalability and often miss model-specific failure modes.

In this work, we introduce CLEAR, a novel interactive tool for AI developers, designed to reduce the overhead of manual error analysis. Our approach utilizes an LLMaJ for generating textual feedback, and conducts discovery of recurring issues via Key Points Analysis (KPA) (Bar-Haim et al. 2020). This method allows us to provide structured, textual feedback that characterizes and quantifies a model's recurring weaknesses and issues across a whole dataset. These insights may guide further improvements, such as prompt engineering, model fine-tuning, or choosing a different LLM.

The CLEAR pipeline is illustrated in Figure 1. It starts with per-instance judgments, which include both a numeric score and textual feedback. It then employs a KPA module to categorize these individual critiques into a concise set of automatically-discovered issues. Each identified issue is mapped back to its matching judgments, which provides quantification for its prevalence, and allows the user to drill down from an issue to its specific examples. Lastly, we provide a user interface that allows for easy and dynamic exploration of issues within the data.

To demonstrate our system's capabilities and significance, we ran CLEAR on several RAG and math benchmarks. We analyzed responses from several systems using different LLM Judges and KPA implementations. We also conducted a user study. Its results confirm the usefulness of CLEAR for real-world AI developers, and its potential value for reducing the time and effort required for error analysis.

Our contributions are: (1) a novel setup for transforming instance-level critiques into system-level issues, (2) an open-source Python package and interface implementing this workflow, and (3) demonstrations across multiple domains, including a user study validating effectiveness.

Method

CLEAR is designed to produce system-level feedback by analyzing a model's behavior across a dataset. The full setup, illustrated in Figure 1, takes as input a dataset of instructions and a target system, and outputs a concise, structured, and quantified summary of the system's recurring issues.

Formally, we assume a dataset $\mathcal{D} = \{x_n\}_{n=1}^{N}$ of N instructions and a target system s, which generates responses $\mathcal{R} = \{r_n\}_{n=1}^{N}$, where each $r_n = s(x_n)$. Our framework then proceeds in two stages:

An LLM-based judge J is prompted to evaluate each pair (x_n, r_n). For each instance, J returns a tuple $j_n = (t_n, s_n)$, where t_n is a natural language critique and s_n is a numeric quality score. These instance-level judgments capture localized failures observed by the judge.

The second stage clusters recurring patterns across the textual feedbacks $\{t_n\}_{n=1}^{N}$ into a set of concise, interpretable

[*]These authors contributed equally.

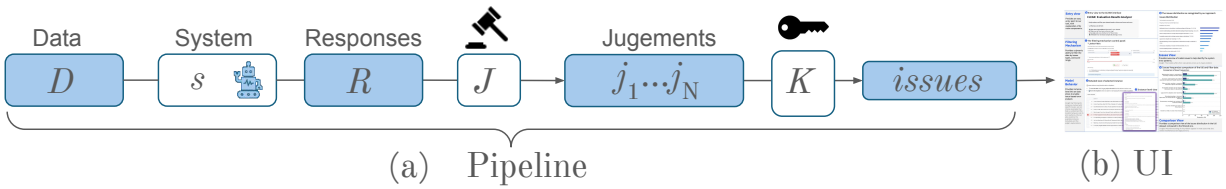

Figure 1: **The CLEAR Framework.** (a) Pipeline- Given a dataset (D) and a target system (s), the system generates responses (R). A judge (J) provides per-instance textual feedback and a score ($\{j_i\}_{i=1}^{N}$). A Key Point Analysis module (K) extracts recurring issues and maps them to the individual j_i's. The discovered issues can be explored via the UI (b).

issues $\{i_m\}_{m=1}^{M}$. For efficiency, and because the focus is on identifying shortcomings, only feedback associated with $s_n < 1$ is considered during issue generation. Each x_n is then linked to one or more relevant issues from this set. We explore two distinct implementations for this aggregation module, denoted K:

Key Point Analysis (KPA): We adopt a classical KPA pipeline. We start by normalizing the critiques into short sentences by an LLM, and then apply the KPA method to cluster the sentences and construct a set of issues over them.

LLM-Based KPA: As an alternative approach, we propose LLM-Based KPA. We summarize each critique t_n, and then prompt an LLM with a batch of these summaries to identify high-level recurring issues. Finally, each t_n is mapped to the derived issue set via a matching prompt. This process requires $\sim 2N$ LLM calls.

CLEAR Framework

Pipeline. To support easy integration and usability, we provide CLEAR as a Python package available on PyPI. The package implements an end-to-end workflow: it generates model responses, evaluates them using an LLM judge, and performs key point analysis to identify recurring issues. Each component in the pipeline can be used independently or in combination, allowing users to customize the workflow to their specific needs or preferences.

Evaluation modes. To accommodate different preferences for issue discovery, CLEAR supports three evaluation modes: *General*, discovering issues dynamically without data-specific prior knowledge, enabling broad, exploratory assessment. *Task-specific*, guiding the judge with domain criteria but allowing additional discoveries. *Static*, mapping evaluation texts to a user-provided predefined list of issues.

User interface. To facilitate exploration, CLEAR provides a visual analytics interface that links high-level error trends with instance-level evidence. An *Issues View* summarizes recurring issues with their frequencies and severities; a *Filtering panel* enables flexible exploration of issue types and score ranges; and a *Comparison View* shows how issue distributions shift under different filters. Finally, users can drill down in the *Instance-Level View*, inspecting instructions, responses, judge feedback, and mapped issues. Together, these synchronized views extend evaluation beyond scalar metrics, helping researchers and practitioners uncover patterns, diagnose errors, and prioritize improvements.

CLEAR: Case Study

Setup. We applied CLEAR to 3 datasets: GSM8K (Cobbe et al. 2021) (math problems) and two RAG benchmarks, TechQA (Castelli et al. 2019) and DelucionQA (Sadat et al. 2023). Systems included Mixtral-8x7B (Jiang et al. 2024), LLaMA-3.1 8B (Grattafiori et al. 2024), Granite-3.3 8B (Granite Team 2024), and Phi-4 (Abdin et al. 2024). Judges were GPT-4o (OpenAI et al. 2024) and LLaMA-3.3 70B, with both IBM Watsonx KPA and our LLM-based variant.

Results. (1) *Actionable insights*- On GSM8K, Mixtral often makes calculation errors and misapplies concepts (e.g., "Mathematical errors in calculations, including rounding in final steps."), suggesting remedies such as training on synthetic reasoning data or pairing with tools like a calculator.

(2) *Dataset-specific issues* – On TechQA, issues include vague technical terms and hallucinated references (e.g., "Lack of clarity in explaining technical details"; "Generates unsupported or speculative information"), surfacing both the dataset's technical focus and known RAG challenges, showing that CLEAR adapts to benchmark characteristics.

(3) *System-level differences* – Comparing Mixtral-8x7B and Phi-4 on TechQA revealed distinct weaknesses: Mixtral often showed "Omission of relevant links or references", while Phi-4 produced fewer errors overall (23.4% vs. 48.1% of instances), aligning with external overall quality metrics.

(4) *Evaluation modes* – On TechQA, task-specific mode increased sensitivity to faithfulness errors, e.g., "Fails to accurately incorporate document information", while general mode revealed novel issues like "Incomplete or abrupt ending of the response".

(5) *KPA method comparison*- Overall, the LLM-based KPA, particularly with GPT-4o, produced more synthesized and specific issue descriptions compared to the traditional KPA implementation (e.g. "incorrect handling of units or conversions" vs. "The calculation in step 7 is incorrect").

User Study. We conducted a user study with 12 AI practitioners and researchers, who explored system outputs through the interface and completed questionnaires. Participants rated the tool highly for usefulness (4.33), time savings (4.25), and surfacing overlooked issues (4.16), all on a 1-5 Likert scale. Most reported (74%) that they would consider taking action based on the tool's output. The results confirmed the tool's value, highlighting CLEAR usefulness for error detection automation.

References

Abdin, M.; Aneja, J.; Behl, H.; Bubeck, S.; Eldan, R.; Gunasekar, S.; Harrison, M.; Hewett, R. J.; Javaheripi, M.; Kauffmann, P.; Lee, J. R.; Lee, Y. T.; Li, Y.; Liu, W.; Mendes, C. C. T.; Nguyen, A.; Price, E.; de Rosa, G.; Saarikivi, O.; Salim, A.; Shah, S.; Wang, X.; Ward, R.; Wu, Y.; Yu, D.; Zhang, C.; and Zhang, Y. 2024. Phi-4 Technical Report. arXiv:2412.08905.

Bar-Haim, R.; Eden, L.; Friedman, R.; Kantor, Y.; Lahav, D.; and Slonim, N. 2020. From Arguments to Key Points: Towards Automatic Argument Summarization. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4029–4039. Online: Association for Computational Linguistics.

Castelli, V.; Chakravarti, R.; Dana, S.; Ferritto, A.; Florian, R.; Franz, M.; Garg, D.; Khandelwal, D.; McCarley, S.; McCawley, M.; Nasr, M.; Pan, L.; Pendus, C.; Pitrelli, J.; Pujar, S.; Roukos, S.; Sakrajda, A.; Sil, A.; Uceda-Sosa, R.; Ward, T.; and Zhang, R. 2019. The TechQA Dataset. arXiv:1911.02984.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Gera, A.; Boni, O.; Perlitz, Y.; Bar-Haim, R.; Eden, L.; and Yehudai, A. 2025. JuStRank: Benchmarking LLM Judges for System Ranking. arXiv:2412.09569.

Granite Team, I. 2024. Granite 3.0 Language Models.

Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Hanna, E. B.; Bressand, F.; Lengyel, G.; Bour, G.; Lample, G.; Lavaud, L. R.; Saulnier, L.; Lachaux, M.-A.; Stock, P.; Subramanian, S.; Yang, S.; Antoniak, S.; Scao, T. L.; Gervet, T.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2024. Mixtral of Experts. arXiv:2401.04088.

Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv:2303.16634.

Murahari, V.; Deshpande, A.; Clark, P.; Rajpurohit, T.; Sabharwal, A.; Narasimhan, K.; and Kalyan, A. 2023. Qualeval: Qualitative evaluation for model improvement. *arXiv preprint arXiv:2311.02807*.

OpenAI; :; Hurst, A.; Lerer, A.; Goucher, A. P.; Perelman, A.; Ramesh, A.; Clark, A.; Ostrow, A.; Welihinda, A.; et al. 2024. GPT-4o System Card. arXiv:2410.21276.

Sadat, M.; Zhou, Z.; Lange, L.; Araki, J.; Gundroo, A.; Wang, B.; Menon, R.; Parvez, M.; and Feng, Z. 2023. DelucionQA: Detecting Hallucinations in Domain-specific Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 822–835. Singapore: Association for Computational Linguistics.

Wu, T.; Ribeiro, M. T.; Heer, J.; and Weld, D. S. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 747–763.

Zeng, Z.; Wang, Y.; Hajishirzi, H.; and Koh, P. W. 2025. Evaltree: Profiling language model weaknesses via hierarchical capability trees. *arXiv preprint arXiv:2503.08893*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 46595–46623. Curran Associates, Inc.