CLEAR：基于 LLM 作为判断器的误差分析简化

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

[I]IBM Research [H]The Hebrew University of Jerusalem
{ Asaf.Yehudai, Y.Perlitz } @ibm.com { lilache, roybar, shmueli } @il.ibm.com

Abstract

对于大型语言模型（LLMs）的评估越来越依赖其他 LLMs 作为评判者。然而，当前的评估范式通常仅提供一个单一的分数或排名，回答哪种模型更好，而不是为什么。虽然这些顶级分数对于基准测试至关重要，但它们掩盖了模型性能背后的具体、可执行原因。为了弥补这一差距，我们引入了 CLEAR，这是一个用于基于 LLM 的错误分析的交互式开源软件包。CLEAR 首先生成每个实例的文本反馈，然后创建一组系统级别的错误问题，并量化每个识别出的问题的普遍性。我们的软件包还为用户提供了一个交互式仪表板，允许通过聚合可视化进行全面的错误分析，应用交互式过滤器以隔离特定问题或分数范围，并深入到展示特定行为模式的个别实例。我们展示了 CLEAR 对 RAG 和数学基准的分析，并通过用户案例研究展示了其实用性。代码：https://ibm.biz/CLEAR-code-repo

1 介绍

生成式人工智能系统的评估正在迅速采用 LLM-as-a-Judge（LLMaJ）范式 (Zheng et al., 2023)，其中由 LLM 进行的自动评估补充甚至替代了人类标注者。基于 LLM 的评审通常应用于对 LLM 响应的质量进行评级或评分 (Liu et al., 2023)，或从多个候选项中选择一个偏好的响应。

将这些判断汇总在许多示例中，为 AI 开发者提供了对其系统的稳健评估，同时也对不同系统或模型进行了系统比较和排名 (Gera et al., 2025)。然而，仅有这些分数或评级对模型的行为提供的见解很有限。AI 开发者仍然依赖于繁琐的手动错误分析来识别反复出现的问题，了解其系统的当前局限性，并有效地规划和优先安排下一次改进的迭代。

在这项工作中，我们介绍了 CLEAR，这是一种为 AI 开发人员设计的新型交互工具，旨在减少手动错误分析的开销。我们的方法利用了 LLMaJ 来生成文本反馈，并通过关键点分析（KPA）(Bar-Haim et al., 2020a) 进行重复问题的发现。这种方法允许我们提供结构化的文本反馈，以描述和量化整个数据集中模型反复出现的弱点和问题。这些见解可能会指导进一步的改进，例如提示工程、模型微调或选择不同的 LLM。

CLEAR 流程如图 1 所示。它从每个实例的判断开始，包括数值评分和文本反馈。然后使用 KPA 模块将这些单独的评论分类为一组自动发现的问题。每个识别出的问题都映射回其匹配的判断，这为其普遍性提供量化，并允许用户从问题深入到其具体示例。最后，我们提供了一个用户界面，便于轻松和动态地探索数据中的问题。

为了展示我们系统的能力，我们在几个 RAG（上下文问答）和数学基准上运行了我们的系统。我们使用不同的 LLM Judges 和 KPA 实现分析了几个系统的响应。为了进一步展示我们系统对现实世界中 AI 开发者的可用性，我们还进行了用户研究。其结果证实了 CLEAR 的有用性，以及它在减少错误分析所需时间和精力方面的潜在价值。

我们的工作做出了以下贡献：

1. 我们提出了一种新的设置，通过总结和结构化实例级反馈来生成自动化的系统级问题。

2. 我们展示了 CLEAR，一个实现我们提出方法的开源演示工具，以及一个交互式用户界面，这二者为研究人员和开发人员提供了一种可行的方式，以更深入地了解其模型的行为。

3. 我们在多个领域展示了该系统，并进行了一项用户研究以确认其有效性。

2 方法

CLEAR 被设计用来通过分析模型在数据集上的行为产生系统级反馈。完整的设置如图 1 所示，它将一个指令数据集和一个目标系统作为

*Equal contribution.

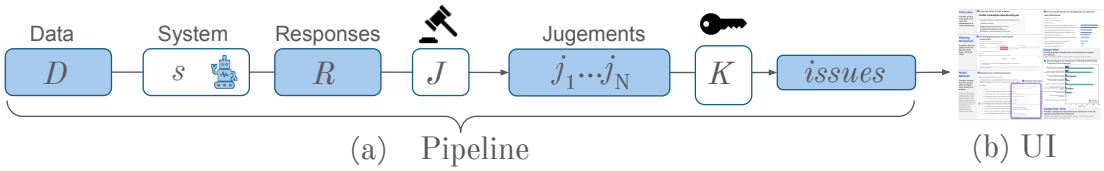

Figure 1: CLEAR 框架。(a) 工作流程- 给定一个数据集 (D) 和一个目标系统 (s)，系统生成响应 (R)。评判者 (J) 为每个实例提供文本反馈和评分 ($\{j_i\}_{i=1}^N$)。关键点分析模块 (K) 提取重复出现的问题并将其映射到各个 j_i 。通过 UI 可以探索发现的问题 (b)。

输入，输出关于系统重复出现问题的简明、结构化和量化的总结。

形式上，我们假设一个包含 N 条指令的数据集 $\mathcal{D} = \{x_n\}_{n=1}^N$ 和一个目标系统 s 。系统生成一组对应的响应 $\mathcal{R} = \{r_n\}_{n=1}^N$ ，其中每个 $r_n = s(x_n)$ 。我们的框架接下来分两个主要阶段进行：首先，一个基于 LLM 的评估者 J 被提示来评估每对 (x_n, r_n) 。对于每个实例，J 返回一个元组 $j_n = (t_n, s_n)$ ，其中 t_n 是自然语言评论，s_n 是数值质量评分。这些实例级别的判断捕捉到评估者观察到的局部失败或优点。我们注意到我们的设置是无参考的，但当有参考可用时，可以用作评估者的上下文。第二阶段把 $\{t_n\}_{n=1}^N$ 中文本反馈中反复出现的模式聚类成一组简明可解释的问题 $\{i_m\}_{m=1}^M$ 。为了提高效率，并因为重点在于识别不足之处，只有与 $s_n < 1$ 相关的反馈在问题生成期间被考虑。每个 t_n 随后被链接到这一组中的一个或多个相关问题。我们探索了两种不同的实现方式用于此聚合模块，记为 K 。

关键点分析 (KPA)：我们采用经典的 KPA 流程 (Bar-Haim et al., 2020a,b) ，它非常适合包含短句子的文本，如论点或产品评论。为提高兼容性，我们首先使用 LLM 将每个 t_n 分解为简短且结构良好的句子。随后，我们应用 KPA 方法对句子进行聚类，并在此基础上构建一组议题。此聚类允许将每个判断映射到其句子所表达的议题上。

基于 LLM 的 KPA：作为一种替代方法，我们提出了基于 LLM 的 KPA。首先，我们通过调用 LLM 将每个评论 t_n 总结为一个更短、更标准化的形式。然后，我们用一批这些总结提示 LLM 以识别高层次的反复出现的问题，并再次用来去除重复并合并最终的问题列表。最后，每个 t_n 通过匹配提示映射到导出的问题集。此过程需要 $\sim 2N$ 次 LLM 调用。实现细节在附录 E 中提供。

3 CLEAR 框架

3.1 流水线

为了支持简单的集成和可用性，我们提供了作为 PyPI 上的一个 Python 包的 CLEAR 。该包实现了一个端到端的工作流程：它生成模型响应，使用 LLM 评估者进行评估，并执行关键点分析以识别反复出现的问题。管道中的每个组件都可以独立使用或结合使用，使用户能够根据其特定需求或偏好定制工作流程。此外，CLEAR 管道设计为高度可配置的。它支持多个推理 API 提供者来生成预测，或者在预测已可用的情况下，可以独立运行评估步骤。如果判断是单独获得的，管道可以直接执行最终评估步骤 (K 步骤)。

为了适应不同的问题发现偏好，CLEAR 支持三种评估模式：(1) 一般模式：使用通用评估提示动态发现问题，能够进行广泛的探索性评估，而无需特定数据的先验知识；(2) 任务特定模式：用户提供特定的问题作为评估标准，引导评审，同时允许进行额外的发现；(3) 静态模式：由用户提供的问题预定义列表作为唯一的评估标准交给评审，直接映射到评估文本中，不进行任何动态发现。

代码示例 CLEAR 可以通过 PyPI 安装：

```
$ pip install clear-eval
```

一旦安装完成，可以用一个 CLI 命令执行完整的分析：

或者使用下面三行 Python 代码：

```
from clear_eval.analysis_runner
    import run_clear_eval_analysis
config_path = <path-to-config>
run_clear_eval_analysis(config_path
    )
```

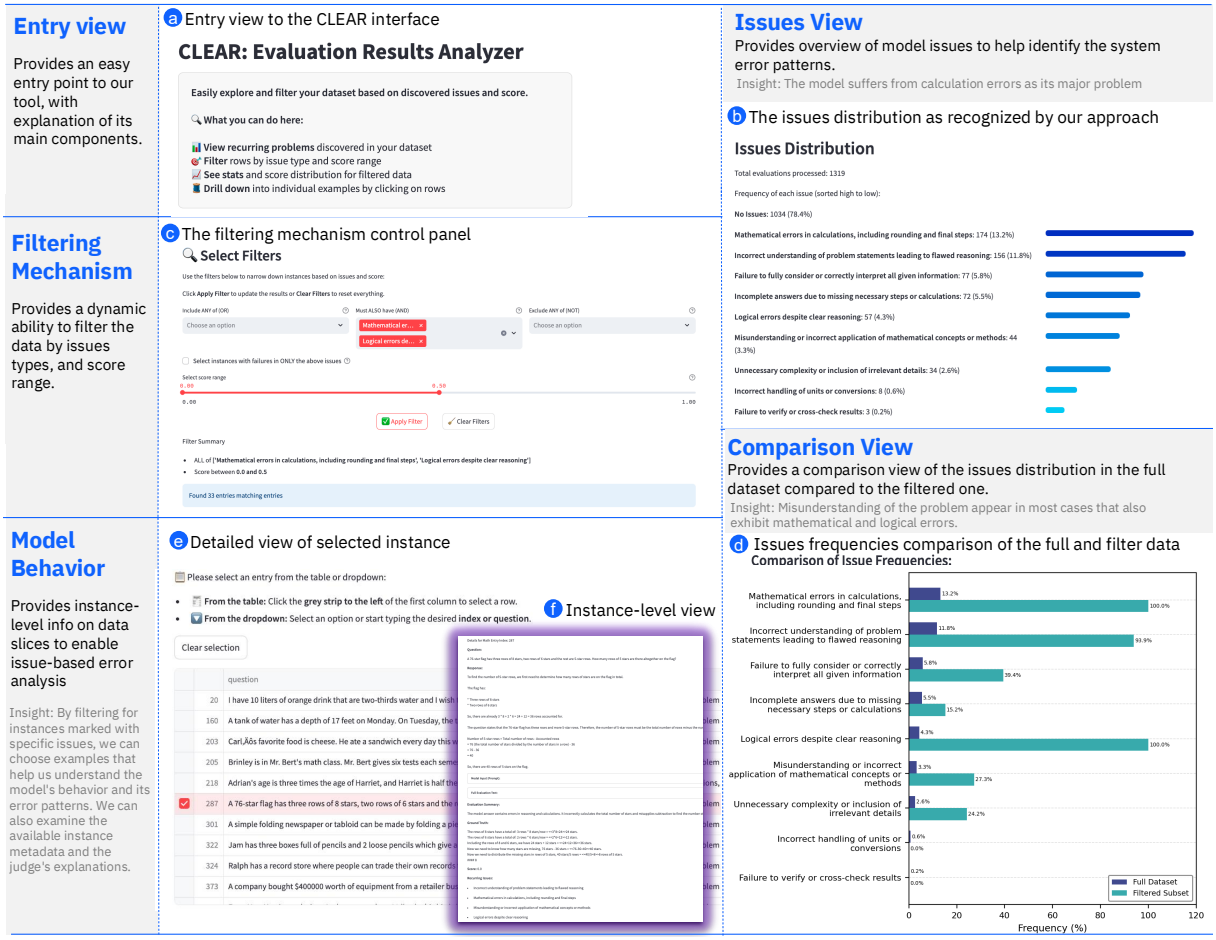

Figure 2: 该图展示了用于分析模型评估结果的 CLEAR 工具的关键组件。(a) 界面的入口，用于探索模型结果和问题。(b) 问题视图可视化检测到的模型错误分布。(c) 过滤机制允许基于问题类型和分数进行过滤以隔离相关示例。(d) 比较视图对比完整数据集和过滤子集之间的问题频率，突出共现模式。(e/f) 模型行为和实例级视图提供详细的、示例级的洞察，以促进细粒度错误分析和模型诊断。

其中 <path_to_config> 是一个 YAML 文件的路径，该文件包含基本配置选项，例如选择的评估器和提供者、输入数据的路径、结果文件夹的路径以及其他可选参数。

一旦处理完成，可以使用下面的 CLI 命令启动 Streamlit 界面。然后，可以通过应用程序手动加载保存到指定输出目录的结果 ZIP 文件。

```
$ run-clear-eval-dashboard
```

3.2 用户界面

为了支持直观且有效地探索模型错误，CLEAR 包含一个为研究人员和实践者设计的可视化分析界面。用户界面（图 2）提供了多个同步视图，结合在一起能全面理解模型行为、错误模式及其分布。

界面由以下关键组件组成：

问题视图。 问题视图（图 2 b）显示了系统识别出的所有问题的概览。每个问题都列出了它在数据集中的出现频率和百分比。这帮助用户一目了然地识别出主要的故障模式。同时，它也帮助用户理解所呈现问题的严重性。

过滤机制。 过滤面板（图 2 c）使用户能够根据特定的问答类型组合或评分范围缩小数据集。我们允许将问题进行并集或交集，或取其反。对于针对性探索这是必不可少的——例如，仅孤立具有逻辑错误的高分答案，或筛选出显示抽取性问题的回答。面板提供动态控制，并且任何应用的过滤器会立即更新其余的视图。

对比视图。 比较视图（图 2 d）可视化了在应用过滤时问题频率的变化。这种比较使用户能够更好地理解不同问题之间，以及问题与分数范围之间的联系。

模型行为和实例级视图。 界面的底部行（图 2 e, f）专注于实例级别的分析。用户可以深入查

看具体的例子，检查原始指令和回应、评委的文字反馈以及其对应的问题列表。这种抽象问题与具体示例之间的直接联系可以帮助用户理解不同问题如何影响他们的系统行为。

综合来看，这些观点使得 CLEAR 成为分析 LLM 输出超越标量指标的工具。该界面支持广泛的模式和细致的检查，帮助实践者发现失败趋势，识别脆弱行为，并更好地理解模型响应在实践中如何失败。

4 CLEAR：案例研究

为了研究我们方法的表现，我们使用了三个数据集：GSM8K (Cobbe et al., 2021) 用于数学文字题，以及两个检索增强生成（RAG）数据集：TechQA (Castelli et al., 2019) 和 DelucionQA (Sadat et al., 2023)，这些数据集基于 RAGBench (Friel et al., 2024) 的处理。我们对四个开放系统进行评估：Mixtral 8x7B (Jiang et al., 2024)，LLaMA-3.1 8B (Grattafiori et al., 2024)，Granite-3.3 8B (Granite Team, 2024)，和 Phi-4 (Abdin et al., 2024)。

我们使用这些系统为每个数据集生成响应，并应用我们的流程来评估它们。对于判断组件，我们在没有参考的情况下使用两种强大的模型，具备通用和任务特定模式。作为一个高质量的闭源评判者，我们使用 GPT-4o (OpenAI et al., 2024)，而作为一个开源替代方案，我们使用 LLaMA-3.3 70B。这些评判者生成的每个实例的反馈通过两种版本的 K 模块：即 IBM watsonx® 关键点分析（KPA）实现[1]，以及我们基于大型语言模型的 KPA，使用 GPT-4o 和 LLaMA-3.3 70B。这个完整的流程为每个系统与数据集对生成一组来源于评判者反馈的重复问题。

4.1 结果

可操作的洞察力。 表 1 展示了在使用特定任务模式评估 GSM8K 时 Mixtral 8x7B 的 CLEAR 结果。结果显示，Mixtral 8x7B 最明显的弱点源于与计算相关的错误。问题还包括错误应用数学概念以及在处理单位或换算时的困难。这些问题有助于检测模型的失败是由于推理缺陷、执行错误，还是两者兼有。重要的是，这一见解是可操作的：开发人员可以选择通过增加专注于数值推理的合成示例来扩充训练数据 (Yehudai et al., 2024)，而用户可能通过将模型与计算器等外部工具配对来弥补这些不足 (Qin et al., 2023)。

在表格的上半部分 2 中，我们展示了在一般模式下对 TechQA 进行评估的 Mixtral 8x7B

[1]IBM watsonx KPA

GSM8K (s : Mixtral 8x7B, J : GPT-4o)
tightitem0
No Issues Detected (78.4%)
Mathematical errors in calculations, including rounding and final steps. (13.2%)
Incorrect understanding of problem statements leading to flawed reasoning. (11.8%)
Failure to fully consider or correctly interpret all given information. (5.8%)
Incomplete answers due to missing necessary steps or calculations. (5.5%)
Logical errors despite clear reasoning. (4.3%)
Misunderstanding or incorrect application of mathematical concepts or methods. (3.3%)
Unnecessary complexity or inclusion of irrelevant details. (2.6%)
Incorrect handling of units or conversions. (0.6%)
Failure to verify or cross-check results. (0.2%)

Table 1: 在具有任务特定模式的 GSM8K 基准测试中针对 Mixtral 8x7B 识别出的问题，按频率降序排列（如括号中所示）。

Mixtral 8x7B (J : GPT-4o)
tightitem0
No Issues Detected (51.9%)
Omission of necessary details or steps (36.3%)
Lack of specificity and completeness in responses (31.2%)
Omission of relevant links or references (9.2%)
Inaccurate or irrelevant information (8.6%)
Failure to provide actionable insights or solutions (8.3%).
Misinterpretation or misuse of context(4.5%)
Lack of clarity in explaining technical details (3.5%)
Incomplete or abrupt ending of the response (3.5%)
Phi-4 (J : GPT-4o)
tightitem0
No Issues Detected (76.6%)
Lacks completeness and necessary details (10.9%)
Lacks context-specific information (9.9%)
Lacks specificity in technical details (6%)
Fails to mention unsupported features or limitations (5.1%)
Inaccurate or fabricated information (2.6%)
Does not directly answer the question (1.9%)
Assumes unsupported or incorrect context (1.9%)

Table 2: 以通用模式在 TechQA 上识别出的 Mixtral 8x7B 和 Phi-4 的问题，按频率递减排序（在括号中显示）

（与 J : GPT-4o）的 CLEAR 结果。问题包括缺失的上下文、技术内容的缺乏具体性以及幻觉或捏造的信息——这是在 RAG 设置中大型语言模型的众所周知的失败模式。值得注意的是，CLEAR 展示了将问题发现适应于任务和数据集

的能力。例如，提取的问题明确反映了 TechQA 对准确、领域特定的技术细节的需求，说明了系统如何根据每个基准的特征来定制反馈。

为了研究 CLEAR 在不同系统中的表现，我们分析了 Mixtral 8x7B 和 Phi-4 在 TechQA 上的结果（见表 2 ）。首先，我们观察到这两个系统产生了不同的问题集，这表明 CLEAR 提供了系统特定的诊断反馈。例如，Mixtral 8x7B 显示出一些独特的问题，比如“遗漏相关链接或参考”和“未能提供可行的见解或解决方案”，而这些在 Phi-4 中没有观察到。

其次，我们发现 Phi-4 被标记实例的整体比例较低，Mixtral 8x7B 为 48.1 %，而 Phi-4 为 23.4 %。这与文献中报道的模型整体质量一致：Phi-4 在 MMLU 评测中获得了 84.8 的分数 (Hendrycks et al., 2021) ，以及在 Chatbot Arena 中获得了 1257 的 Elo 分数 (Chiang et al., 2024) ，而 Mixtral 8x7B 的 MMLU 分数和 Elo 分数均为 1194。这些发现表明，CLEAR 可以支持高层次的系统比较，此外还可以提供细粒度的诊断反馈。

评估模式的影响 我们还比较了任务特定和一般评估模式的使用。我们的研究结果表明，任务特定模式提高了 CLEAR 在与任务紧密相关的问题上的敏感度。相比之下，一般模式倾向于揭示更广泛范围的、更细微或未预料到的问题。例如，在 RAG 数据集上，任务特定提示帮助揭示了更多与忠实性相关的问题，如“生成不支持或推测性信息”，而这些问题在一般提示中被遗漏了。另一方面，一般模式发现了更多新颖的问题，比如“回复的不完整或突然结束”（详见附录 C 的完整对比）。

最后，我们对 K 模块的三种实现进行了关键点分析的定量评估。我们的结果表明，基于 LLM 的 KPA 往往产生的问题比传统的 KPA 方法更具合成性和较少提取性。在评估的模型中，GPT-4o 产生的问题类型比 LLaMA-3.3 和 Watsonx 的实现更加准确和具体（见附录 B ）。

4.2 用户研究

为了评估 CLEAR 对用户的有用性和可用性，我们进行了一个包含 12 位 AI 从业者和研究人员的用户研究。参与者被要求在三个数据集上使用该工具，探索界面，并通过结构化问卷（使用李克特量表）和自由评论提供反馈。评估的维度包括工具的有用性、与他们当前实践的比较以及他们对工具的信任度。（详见附录 D ，了解参与者、说明和问题的详细描述）。

结果表明，用户认为 CLEAR 对于表面层分析很有价值。具体来说，参与者特别认可其错误检测的自动化（指出 75% 目前在他们的使用情况下依赖人工检查）、视觉探索界面以及发现他们可能忽略的问题的潜力，平均评分为 4.33 。该系统被视为可操作的，74% 的参与者报告他们会根据输出采取或考虑采取行动，省时且优于现有实践，这两个方面的平均得分都是 4.25。特别是在大规模识别常见故障模式方面被高度评价，平均得分为 4.16。尽管整体上得到积极的反馈，用户仍指出了一些需要改进的地方。若干反馈表达了对问题可信度、3.83 评分及具体性的疑虑。有些用户觉得描述含糊不清，或难以判断哪些错误最为严重。用户还请求提供例如严重性注释、更清晰的分类、自动摘要以及更好的文本反馈高亮等功能。

5 相关工作

现有的错误分析工具通过检查数据集实例来映射模型错误和能力，例如 EvalTree 的 (Zeng et al., 2025) 能力层次结构或 Qualeval 的 (Murahari et al., 2023) 数据属性。其他方法是交互式的，比如 Erudite (Wu et al., 2019) ，需要用户标记来对错误进行聚类，或者使用像 MisattributionLLM (Xu et al., 2025) 这样的专业模型来评分已知错误类型。

关键是，所有这些方法都依赖于有标签的数据，因此限制了其在特定任务上的使用。此外，由于这些研究是基于数据集特征来探测弱点或能力，而不是基于模型特定行为，因此它们可能会忽略模型特有的故障模式。

6 结论与未来工作

我们展示了 CLEAR ，这是一种新颖的框架和交互工具，用于自动化生成式 AI 系统的错误分析。通过利用 LLMaJ 评估和关键点分析，CLEAR 从实例级批评中提取结构化的系统级反馈。这使得 AI 开发人员能够超越标量指标，并在最小化手动开销的情况下，发现反复出现且可操作的失败模式。

我们的实验涵盖了数学和 RAG 数据集，表明了 CLEAR 对不同任务和模型的适应能力，揭示了既有普遍性又有系统特定的问题。该工具在评估模式上提供了灵活性，支持多个评委和 KPA 配置，并包括用于深入探索模型行为的直观可视界面。我们的用户研究证实了该工具对于从业者的价值，突显其节省时间、提供新见解和改进分析工作流程的能力，但也指出了需要进一步增强的领域。

展望未来，我们计划提高发现问题的具体性和清晰度，加入严重性评分和优先级机制，并探索增加用户信任和可解释性的方法。此外，我们还旨在集成交互式反馈循环，允许用户细化或纠正发现的问题。

CLEAR 是公开可获取的，我们希望它能够作为社区的一块垫脚石，推动生成模型更透明、高效和富有洞察力的评估。

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. 2024. Phi-4 technical report. *Preprint*, arXiv:2412.08905.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 39–49, Online. Association for Computational Linguistics.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Mike McCawley, Mohamed Nasr, Lin Pan, Cezar Pendus, John Pitrelli, Saurabh Pujar, Salim Roukos, Andrzej Sakrajda, Avirup Sil, Rosario Uceda-Sosa, Todd Ward, and Rong Zhang. 2019. The techqa dataset. *Preprint*, arXiv:1911.02984.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*.

Ariel Gera, Odellia Boni, Yotam Perlitz, Roy Bar-Haim, Lilach Eden, and Asaf Yehudai. 2025. Justrank: Benchmarking llm judges for system ranking. *Preprint*, arXiv:2412.09569.

IBM Granite Team. 2024. Granite 3.0 language models.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *Preprint*, arXiv:2401.04088.

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. *Preprint*, arXiv:2303.16634.

Vishvak Murahari, Ameet Deshpande, Peter Clark, Tanmay Rajpurohit, Ashish Sabharwal, Karthik Narasimhan, and Ashwin Kalyan. 2023. Qualeval: Qualitative evaluation for model improvement. *arXiv preprint arXiv:2311.02807*.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, et al. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Mobashir Sadat, Zhengyu Zhou, Lukas Lange, Jun Araki, Arsalan Gundroo, Bingqing Wang, Rakesh Menon, Md Parvez, and Zhe Feng. 2023. DelucionQA: Detecting hallucinations in domain-specific question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 822–835, Singapore. Association for Computational Linguistics.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2019. Errudite: Scalable, reproducible, and testable error analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 747–763.

Zishan Xu, Shuyi Xie, Shupei Xiao, Linlin Song, Sui Wenjuan, Fan Lin, and Lv Qingsong. 2025. MisattributionLLM: Integrating error attribution capability into LLM evaluation.

Asaf Yehudai, Boaz Carmeli, Yosi Mass, Ofir Arviv, Nathaniel Mills, Assaf Toledo, Eyal Shnarch, and Leshem Choshen. 2024. Genie: Achieving human parity in content-grounded datasets generation. *Preprint*, arXiv:2401.14367.

Zhiyuan Zeng, Yizhong Wang, Hannaneh Hajishirzi, and Pang Wei Koh. 2025. Evaltree: Profiling language model weaknesses via hierarchical capability trees. *arXiv preprint arXiv:2503.08893*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A 局限性

我们框架的有效性在根本上取决于其组成模型的质量。分析流程继承了底层 LLM-as-a-Judge（J）和关键点分析（K）模块的偏见和潜在不准确性。

对法官质量的依赖 发现的问题的可靠性取决于来自裁判模型的最初批评的可靠性。裁判中的偏见（例如，自身偏见，长度/风格偏见）或未能识别出细微错误都直接影响了最终聚合问题的质量和有效性。

尽管基于 LLM 的 KPA 方法有效，但从计算角度来看可能是昂贵的，对于尺寸为 N 的数据集，需要大约 $\sim 2N$ 次 LLM 调用。然而，我们仅将基于 LLM 的 KPA 应用于初始评估分数较低的实例，这使得成本取决于目标系统的质量。不过，这种开销可能成为大规模分析的实际限制。

缺乏因果性 我们的工具可以识别和量化重复出现的错误模式，但不能诊断其根本原因。例如，它可以突出显示常见的事实不准确性，但不能区分这是否源于知识缺陷、检索失败或有缺陷的推理。

B 方法比较

表 3 对三种 KPA 实现生成的关键点进行了定性比较：Watsonx-KPA、基于 LLM 的 LLaMA-3 KPA 和基于 LLM 的 GPT-4o KPA。虽然这三者都旨在识别反复出现的系统级问题，但它们在抽象层次、措辞风格和最终关键点的普遍性方面存在差异。下面，我们描述了在这些方法中观察到的两个主要变化维度。

抽取式与合成式风格 Watsonx-KPA 方法生成的关键点具有很高的提取性和细粒度，通常几乎直接从原始反馈中提取。因此，其问题往往过于具体（例如，“第 7 步中的计算不正确”）或与特定实例结构相关。虽然这种精确性可以发现具体问题，但它降低了普遍性，且常导致一份长长的、范围狭窄的问题清单。相比之下，基于 LLM 的 KPA——尤其是使用 GPT-4o——生成的关键点则更加抽象和综合。这些关键点将多次出现的相关错误聚合成更广泛的类别，例如“未全面考虑或正确解读所有给定信息”，从而促进在不同例子和系统间的更好泛化。

问题的粒度和清晰度。 基于 LLM 的方法在抽象和清晰度的平衡上也有所不同。基于 LLaMA-3 的 KPA 有时会生成范围广泛的长而复合的关键点（例如，“未能考虑所有相关变量、条件或情景”），这可能会降低可读性并引

入冗余。而 GPT-4o 则倾向于生成简洁且结构良好的关键点，达到了信息性与清晰度之间的平衡（例如，“处理单位或转换错误”）。相比之下，由于 Watsonx 的输出通常与具体实例挂钩，给人一种片段化或重复的感觉，常常导致多个关键点覆盖同一潜在问题的重叠方面。

C 评估模式的影响

表 4 展示了使用一般评价模式与任务特定评价模式在 TechQA 基准上发现的关于 mixtral 8x7b 的所有问题清单，如在 §4.1 中的评估模式影响段落中讨论的那样。

D 用户研究详情

在 12 名参与者中，7 名是应用开发者，3 名是业务分析师，1 名是模型开发者。

为了为研究设置背景，我们从该工具的简要概述开始，然后是逐步的使用说明（见图 3）。

这些问题包括 1-5 的李克特量表、多项选择和开放式格式，分为三个部分以评估该工具的不同方面：

1. 有用性 - 本节探讨该工具在理解模型错误、节省时间或影响调试或开发决策方面的帮助程度（图 4）。
2. 比较价值 - 本节将工具与您当前的方法进行比较——人工检查或现有工具——并帮助识别其增加（或缺乏）价值的领域（图 5）。
3. 信任 & 的可靠性- 本节评估您对该工具输出的信任程度，以及它是否增强了您对模型行为理解的信心（图 6）。

E 实现细节

对于响应生成，所有模型都使用默认参数提示。在所有评估阶段，推理是在温度为 0 的情况下进行的。系统设置为每次分析生成 3 到 15 个关键点。问题合成使用最多 150 个非完美得分的评估总结进行。在 Git Repo 中提供了所有提示。

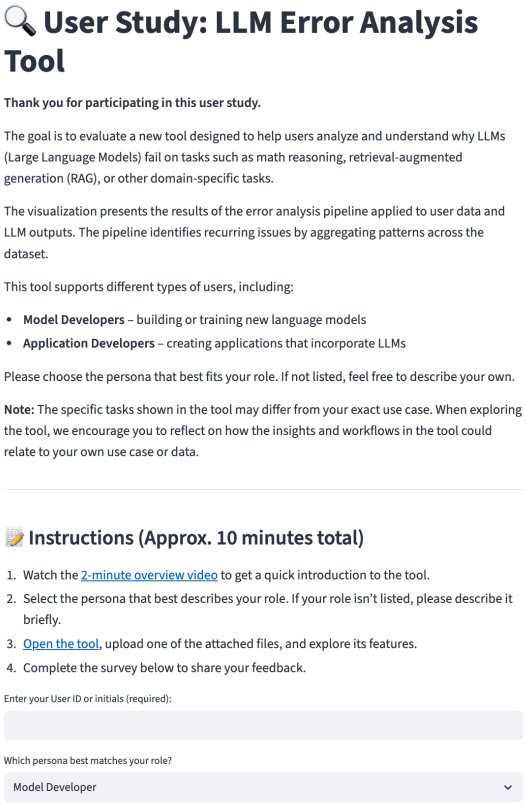

Figure 3: 研究参与者的说明。

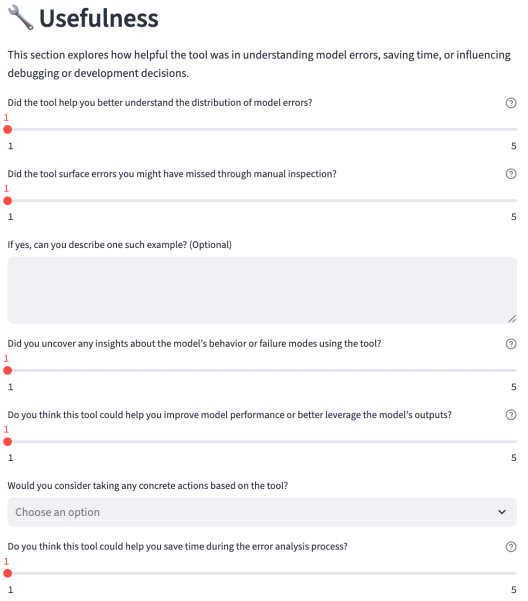

Figure 4: 第 1 节 - 有用性问题。

Dataset	watsonx-KPA	LLM-based (GPT-4o)	LLM-based (LLaMA-3-70B)
GSM8K	tightitem0 No Issues Detected (78.3%) The error leads to an incorrect final answer. (17.8%) The equations are set up inaccurately. (14.4%) The model fails to provide correct reasoning. (12.3%) The calculation in step 7 is incorrect. (11.2%) The error leads to an incorrect average. (10.5%) However, the rounding error could be misleading. (7.1%) It also lacks clarity. (6.3%) It does not verify calculations. (5.6%) The steps in the model answer are incomplete. (5.4%) The model incorrectly calculates the number of pairs. (4.5%) Necessary steps to solve the problem are missing. (3.9%)	tightitem0 No Issues Detected (83.7%) Incorrect understanding of problem statements leading to flawed reasoning (7.5%) Mathematical errors in calculations, including rounding and final steps (7.4%) Incomplete answers due to missing necessary steps or calculations (4.1%) Failure to fully consider or correctly interpret all given information (3.9%) Unnecessary complexity or inclusion of irrelevant details (1.7%) Logical errors despite clear reasoning (1.1%) Incorrect handling of units or conversions (0.4%) Misunderstanding or incorrect application of mathematical concepts or methods (0.3%) Failure to verify or cross-check results (0.2%)	tightitem0 No Issues Detected (81.4 Calculation errors or inaccuracies (15.2%) Flawed reasoning, logical errors, or incorrect application of formulas/algorithms (7.1%) Incorrect assumptions or misinterpretations of problem statements (5.2%) Failure to account for all relevant variables, conditions, or scenarios (5.2%) Lack of clarity, consistency, or unnecessary complexity in explanations (3.1%) Inability to correctly interpret or apply given information (2.9%)
TechQA	tightitem0 No Issues Detected (4.5%) The response lacks completeness and clarity. (89.5%) The model answer lacks relevance and factual support. (70.7%) It fails to provide document-supported solutions. (59.2%) The model misses key details about the vulnerability. (55.1%) As a result, the response lacks faithfulness. (54.8%) It contains unnecessary details and inaccuracies. (41.4%) It misses several necessary troubleshooting steps. (26.4%) The model lacks actionable steps and insights. (21.3%) However, it lacks specific configuration steps for ITCAM. (8.6%) However, the additional join information is unnecessary. (6.4%) The answer misses specific details about cache synchronization. (3.8%)	tightitem0 No Issues Detected (18.5%) Lacks completeness and omits crucial details (59.6%) Generates unsupported or speculative information (31.8%) Fails to accurately incorporate document information (22.0%) Provides irrelevant or extraneous information (14.3%) Lacks clarity and conciseness (14.0%) Fails to address the specific question (12.4%) Fails to provide direct solutions (8.0%) Lacks structured presentation (3.8%)	tightitem0 No Issues Detected 26 (8.3%) Lack of detail and clarity in answers (67.8%) Inadequate consideration of context and user needs (45.9%) Failure to directly address the user's question (43.3%) Unclear or incomplete solutions (36.6%) Lack of faithfulness to original documents (26.8%) Introduction of unnecessary information (24.2%) Failure to provide supporting evidence or examples (22.3%) Failure to provide clear steps or instructions (19.7%) Lack of relevance to the specific question or topic (19.4%) Inaccuracies and inconsistencies in information (15.3%) Overreliance on general knowledge (14.0%) Failure to consider multiple possible causes or solutions (8.3%) Inadequate explanation of technical terms (2.2%)

Table 3: 针对 Mixtral 8x7B 的主要问题，通过 GSM8K 和 TechQA 基准测试的每种方法，采用任务特定的评估模式识别出来。

General (J : GPT-4o)
tightitem0
No Issues Detected (51.9%)
Omission of necessary details or steps (36.3%)
Lack of specificity and completeness in responses (31.2%)
Omission of relevant links or references (9.2%)
Inaccurate or irrelevant information (8.6%)
Failure to provide actionable insights or solutions (8.3%)
Misinterpretation or misuse of context(4.5%)
Lack of clarity in explaining technical details (3.5%)
Incomplete or abrupt ending of the response (3.5%)
Task-Specific (J : GPT-4o)
tightitem0
No Issues Detected (18.5%)
Lacks completeness and omits crucial details (59.6%)
Generates unsupported or speculative information (31.8%)
Fails to accurately incorporate document information (22.0%)
Provides irrelevant or extraneous information (14.3%)
Lacks clarity and conciseness (14.0%)
Fails to address the specific question (12.4%)
Fails to provide direct solutions (8.0%)
Lacks structured presentation (3.8%)

Table 4: 在 TechQA 上为 Mixtral 8x7B 识别的问题，使用通用（上）和任务特定（下）评估模式。

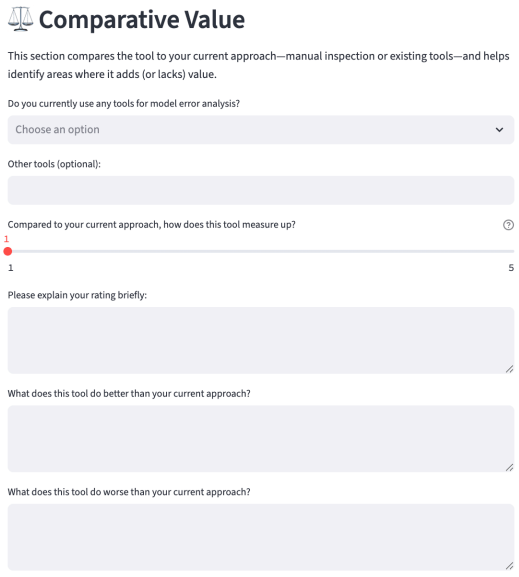

Figure 5: 第 2 节- 比较价值问题。

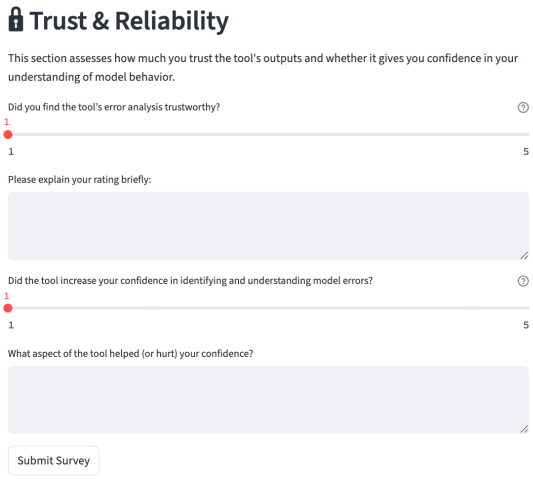

Figure 6: 第 3 节-信任 & 可靠性问题。