Explainable Benchmarking of LLM API Services: Predictive Quality Assessment, Failure Attribution, and Robustness Analysis Under Adversarial Conditions

Milos Carpenter
Department of Computer Science, Binghamton University, Binghamton, NY, USA.
milosc@binghamton.edu

Leisheng Cui
Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, USA.
leishengmail@uab.edu

Madhav Balhotra
Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, USA.
madhavmalhotra595@ku.edu

Abstract

The rapid proliferation of Large Language Model (LLM) application programming interfaces (APIs) as foundational components in socio-technical systems has created an urgent need for rigorous, interpretable, and operationally relevant benchmarking frameworks. Traditional performance evaluations, which emphasize aggregate metrics such as accuracy and latency, fail to capture the nuanced failure modes, predictive degradations, and adversarial vulnerabilities that characterize real-world LLM API deployments. This paper proposes an explainable benchmarking paradigm that integrates predictive quality assessment, systematic failure attribution, and adversarial robustness analysis into a unified evaluation architecture. We ground our approach in structural trade-offs between model fidelity, computational cost, and interpretability, drawing on causal inference and post-hoc explanation methods to attribute API failures to specific model components, input perturbations, or infrastructure bottlenecks. A layered governance framework is introduced to balance transparency, fairness, and sustainability in LLM API provisioning. Through cross-domain analysis of deployment scenarios in healthcare, finance, and content moderation, we illustrate how explainable benchmarking can inform policy decisions, improve system resilience, and foster accountability among API providers. The paper concludes with forward-looking recommendations for integrating predictive quality monitors, failure attribution dashboards, and adversarial testing protocols into continuous deployment pipelines, thereby advancing the reliability and trustworthiness of LLM-based services.

Keywords

explainable AI, LLM API services, benchmarking, adversarial robustness, failure attribution, predictive quality assessment, socio-technical systems, governance.

1. Introduction

The deployment of Large Language Models (LLMs) through application programming interfaces (APIs) has transformed the landscape of artificial intelligence services, enabling organizations to integrate advanced natural language understanding and generation capabilities without bearing the full cost of model training and infrastructure maintenance. These API services now underpin critical applications in healthcare diagnostics, financial analytics, legal document review, customer service automation, and content moderation. As their adoption deepens, the need for systematic, interpretable, and operationally relevant benchmarking becomes paramount. Traditional benchmark suites, such as those based on static datasets and aggregate accuracy metrics, provide a coarse picture of model performance but largely ignore the dynamic, context-dependent nature of real-world usage. They fail to account for the subtle degradation of output quality under shifting input distributions, the latent failure modes that arise from model biases or infrastructure instabilities, and the vulnerability of these systems to adversarial manipulation. This paper addresses these gaps by proposing an explainable benchmarking framework that encompasses three interrelated dimensions: predictive quality assessment, systematic failure attribution, and adversarial robustness analysis.

The motivation for such a framework stems from the observation that LLM API services operate as socio-technical infrastructures where technical performance, economic incentives, and governance mechanisms are deeply intertwined. A purely accuracy-centric evaluation overlooks the fact that a model may produce high-quality outputs on average yet exhibit catastrophic failures in minority subpopulations or under specific adversarial conditions. Moreover, the black-box nature of state-of-the-art LLMs, coupled with the proprietary access restrictions typical of commercial APIs, complicates the identification of failure causes. Without transparent attribution mechanisms, system operators cannot distinguish between failures induced by model limitations, input noise, API rate limiting, or downstream application logic. This opacity undermines trust, hinders iterative improvement, and exposes organizations to significant operational and reputational risks.

Our approach centers on explainability as a first-class citizen in benchmarking. By integrating predictive models of output quality that can forecast degradation before it impacts end-users, we enable proactive intervention. Failure attribution techniques, drawing on interpretable feature attribution and counterfactual reasoning, allow stakeholders to pinpoint root causes. Adversarial robustness evaluation, conducted under realistic threat models, assesses the system's resilience to deliberate attacks intended to manipulate outputs or extract sensitive information. Together, these components form a holistic assessment methodology that informs architectural decisions, deployment governance, and policy development.

This paper is organized as follows. Section 2 surveys related work in LLM evaluation, explainability, and adversarial robustness, identifying the limitations of existing approaches. Section 3 develops a predictive quality assessment model grounded in least squares vector machines and SHAP interpretability, drawing on recent work in API response quality prediction [1]. Section 4 addresses failure attribution, leveraging prototype consistency and vertical split learning defenses [2] to attribute failures to model components or input perturbations. Section 5 examines robustness under adversarial conditions, including prompt injection and evasion attacks, and discusses mitigation strategies from the perspective of multi-agent reinforcement learning for vulnerability patching [3]. Section 6 synthesizes these findings into a discussion of structural trade-offs, governance implications, and sustainability

considerations, with reference to large-scale system design principles [4]. Section 7 concludes with future research directions and policy recommendations.

2. Related Work and Background

The evaluation of LLM API services has traditionally relied on benchmark datasets such as GLUE, SuperGLUE, and HELM, which measure performance on standardized tasks using accuracy, F1 score, and other aggregate metrics. While these benchmarks provide a useful point of comparison, they suffer from several shortcomings. First, they are static and do not capture the distribution shifts that occur when APIs are deployed in different domains or user populations. Second, they ignore the cost-accuracy trade-off inherent in API usage, where users may select different model tiers or prompting strategies. Third, they provide no insight into failure causes or robustness under attack. Recent efforts to introduce more dynamic evaluation paradigms include continuous benchmarking platforms like LMSys Chatbot Arena, which crowdsource human preferences but lack systematic attribution mechanisms.

Explainability in LLMs has been explored through methods such as attention visualization, SHAP values, and integrated gradients. However, these techniques are typically applied to individual predictions rather than to the systemic evaluation of API services. In the context of API benchmarking, explainability must be operationalized at multiple levels: the model level (e.g., which input features drive predictions), the infrastructure level (e.g., how latency or throughput correlates with quality), and the governance level (e.g., which policy settings affect fairness). Recent work on predictive quality assessment using least squares vector machines with SHAP interpretability [1] demonstrates a promising approach for forecasting API response quality and identifying the contributing factors. Similarly, prototype-based defense mechanisms for vertical split learning [2] offer a method for attributing failures to specific model shards or data silos, which is relevant for distributed API architectures.

Adversarial robustness of LLMs has become a critical concern, especially with the rise of jailbreaking attacks, prompt injection, and extraction of training data. Early studies focused on text classification tasks, but recent work extends to generative models. The challenge is even greater for API services, where the adversary has only query access and may exploit rate limiting, response generation latency, or side channels. Robustness analysis must therefore encompass both model-level and system-level vulnerabilities. Multi-agent reinforcement learning approaches for vulnerability patch planning [3] have been proposed to prioritize patches in microservice architectures, which can be adapted to LLM API deployments. Finally, the broader literature on socio-technical systems [4] underscores the importance of aligning technical evaluation with governance structures, fairness constraints, and sustainability goals.

3. Predictive Quality Assessment of LLM API Services

Predictive quality assessment aims to estimate the expected output quality of an LLM API response before the user consumes it, enabling real-time mitigation strategies such as switching to a different model version, re-querying, or flagging the output for human review. Traditional approaches rely on post-hoc evaluation, which is inherently reactive. Our framework adopts a proactive stance by constructing a predictive model of quality using system-level features such as prompt complexity, model temperature, API latency, token count, and contextual similarity to known high-quality exemplars.

Drawing on recent advances in response quality prediction [1], we employ a least squares vector machine regressor trained on a large corpus of LLM API interactions with human-

annotated quality scores. The model is not designed to replace human judgment but to provide a calibrated probability of degradation per query. The key innovation is the integration of SHAP interpretability, which allows us to decompose each prediction into feature contributions. For instance, SHAP analysis reveals that extreme prompt length and high temperature settings are the most influential predictors of quality drops, while API latency serves as a proxy for server-side resource contention. This interpretability is crucial for system operators who need to understand why the predictive model flags a particular request. It also enables the construction of personalized quality thresholds: a financial compliance application may require higher confidence than a casual chatbot.

The predictive model itself is part of a larger monitoring infrastructure that collects streaming logs from the API gateway, extracts feature vectors in real time, and feeds them into the regressor. The output is a continuous quality score and an ordered list of contributing factors. This information can be visualized in a dashboard that highlights anomalous requests and triggers alerts when the predicted quality falls below a configurable threshold. Importantly, the predictive model must be periodically retrained to adapt to shifts in the API's underlying model (e.g., model version updates) and changes in user behavior. The retraining process itself should be auditable, and the SHAP explanations should be made available to downstream stakeholders to ensure transparency.

We also consider the trade-off between prediction accuracy and computational overhead. A lightweight model can be deployed at the edge near the API gateway, while a more complex ensemble can run offline for batch analysis. The choice depends on the latency tolerance of the application: for real-time medical triage, even a few milliseconds of added delay may be unacceptable, whereas for offline content moderation, deeper analysis is justified. This architectural flexibility is a core design principle of our benchmarking framework.

4. Failure Attribution in API-Based LLM Systems

When an LLM API produces a low-quality or harmful output, understanding why is essential for corrective action and accountability. Failure attribution in this context is complicated by the distributed nature of API systems: the root cause may lie in the model weights, the prompt construction, the retrieval pipeline, the infrastructure load, or even the client's network configuration. We propose a multi-level attribution methodology that combines prototype-based clustering, counterfactual reasoning, and causal graph analysis.

Building on the prototype consistency defense for vertical split learning [2], we first construct a set of prototype inputs that correspond to high-quality and low-quality output categories. These prototypes are derived from historical interactions and can be thought of as canonical examples of success and failure. For a given query, we compute the distance (in an embedding space) between the query and each prototype. If the query is closer to a failure prototype, the model's output is likely to be similar. This provides an immediate, interpretable attribution: the failure is associated with a specific pattern captured by the prototype. For example, a prototype might encode "ambiguous instructions followed by hallucinated numerical values." If a new query matches that prototype, the system operator can inspect the similar historical failures and their resolutions.

However, prototype matching alone may not reveal the causal mechanism. We therefore augment it with counterfactual simulation. For each attributed failure, we systematically modify features of the input prompt (e.g., rephrasing ambiguous clauses, adding constraints, removing sensitive terms) and re-query the API to observe whether the output quality

improves. The set of modifications that lead to improvement constitutes a minimal set of changes needed to avoid failure. This approach not only attributes the failure to particular input features but also suggests actionable mitigations. For instance, counterfactual analysis may show that replacing a domain-specific acronym with its full name raises the quality score above the threshold, indicating a failure due to lexical out-of-distribution input.

At the infrastructure level, we incorporate causal graphs that model the relationships between observable variables: API response time, token throughput, server load, error codes, and output quality. Using techniques from structural causal modeling, we can estimate the effect of a rate-limiting incident on downstream quality. This is particularly important for API services where failures may be intermittent and correlated with peak traffic. The causal graph also enables us to attribute blame to specific components in a microservice architecture, such as a load balancer misconfiguration or a model inference node experiencing memory pressure. The output of this attribution step feeds into automated incident response systems and informs the prioritization of patches [3].

5. Robustness Analysis Under Adversarial Conditions

LLM API services face a diverse threat landscape that includes prompt injection, jailbreaking, data extraction, and denial-of-service attacks. Adversaries may attempt to manipulate the system to generate harmful content, leak private information, or degrade service availability. Traditional robustness evaluation focuses on model-level defenses, but a system-level perspective is necessary because the API introduces additional attack surfaces: the network layer, the authentication mechanism, the pricing policy, and the caching infrastructure. Our robustness analysis framework integrates both model-level and system-level adversarial testing, with a focus on explainability of vulnerabilities.

We adopt a taxonomy of adversarial conditions relevant to LLM APIs: evasion attacks (crafting prompts that bypass content filters), poisoning attacks (injecting malicious samples into the training data via feedback loops), and extraction attacks (inferring sensitive information from model responses). For each category, we design a set of adversarial test cases generated using gradient-based search and reinforcement learning. The tests are executed against the API in a sandboxed environment, and the results are recorded alongside contextual metadata such as input length, response time, and error codes. The key contribution is the integration of explanation methods into the adversarial evaluation: after detecting a successful attack, we use SHAP-based feature attribution to identify which elements of the adversarial prompt contributed most to bypassing the filter. This allows defenders to understand the weaknesses in the content moderation pipeline and to design targeted countermeasures.

Furthermore, robustness analysis must account for the dynamic nature of API deployments. Model updates, feature rollouts, and configuration changes can introduce new vulnerabilities or patch existing ones. We propose a continuous adversarial testing pipeline that runs a suite of attacks on every deployment update, with results compared to a baseline. The pipeline outputs a robustness score and a ranked list of vulnerabilities, each annotated with the expected impact and the ease of exploitation. This information feeds into a multi-agent reinforcement learning system for patch planning [3], which optimizes the order of vulnerability fixes given resource constraints. The agents learn to balance the cost of downtime, the severity of the vulnerability, and the risk of adversarial exploitation.

Finally, we discuss the ethical implications of adversarial testing. Responsible disclosure of vulnerabilities to API providers is essential to avoid enabling malicious actors. Our framework includes a governance component that defines protocols for reporting findings and coordinating patches. The goal is to create a cooperative ecosystem where benchmarks are used to improve security rather than to exploit weaknesses.

6. Structural Trade-offs, Governance, and Sustainability

The explainable benchmarking framework outlined in the preceding sections imposes certain structural trade-offs that must be carefully managed. Predictive quality assessment adds computational overhead and may introduce false positives, causing unnecessary re-queries or human review. Failure attribution, particularly counterfactual simulation, can be time-consuming and costly if applied to every low-quality output. Adversarial robustness testing may degrade the user experience if performed in live environments. These trade-offs are not merely technical but also economic and organizational. Organizations must decide how much to invest in monitoring and explainability relative to the expected cost of failures. Our framework provides a cost-benefit analysis tool that quantifies the value of transparency in terms of reduced incident response time, improved model iteration speed, and enhanced user trust.

Governance structures play a critical role in operationalizing explainable benchmarking. We propose a layered governance model consisting of three tiers: the API provider, the intermediary platform, and the end-user organization. At the provider level, governance includes publishing explainability reports, disclosing known failure modes, and offering contractual guarantees on robustness metrics. At the intermediary level (e.g., cloud service platforms that aggregate multiple LLM APIs), governance involves transparent routing decisions based on quality predictions and fairness constraints. At the end-user level, governance requires the ability to audit predictions and to contest decisions based on attribution results. This layered approach aligns with emerging regulatory frameworks such as the EU AI Act, which mandates transparency and risk management for high-risk AI systems.

Sustainability is another dimension that is often overlooked in benchmarking discourse. The computational resources required for LLM API inference are substantial, and the added overhead of explainable monitoring compounds energy consumption. Our framework incorporates sustainability metrics such as total energy per high-quality response, carbon intensity of the hosting region, and efficiency of the predictive model. We argue that a sustainable benchmarking approach must balance the benefits of transparency against the environmental cost. For instance, lightweight predictive models that run on edge devices can reduce the load on central servers while still providing meaningful quality estimates. Similarly, failure attribution can be limited to a stratified sample of low-quality outputs rather than all outputs, thereby reducing computation.

7. Conclusion

This paper has presented a comprehensive framework for explainable benchmarking of LLM API services, integrating predictive quality assessment, failure attribution, and adversarial robustness analysis. By grounding evaluation in interpretable methods and system-level thinking, we address the limitations of traditional accuracy-centric benchmarks. The framework enables proactive quality management, root-cause diagnosis of failures, and continuous monitoring of adversarial resilience. We have discussed the underlying

architectural trade-offs, governance implications, and sustainability considerations that must be addressed for real-world deployment.

Future research should focus on empirically validating the framework across multiple commercial LLM API providers, developing standardized explainability metrics for benchmarking reports, and exploring the integration of human-in-the-loop feedback for refinement of predictive models. Additionally, the extension of failure attribution to multi-model orchestration scenarios, where outputs from several APIs are combined, presents a rich area for investigation. As LLM APIs become further embedded in critical infrastructure, the need for transparent, robust, and accountable evaluation will only intensify. We hope that the principles outlined here contribute to the responsible evolution of this transformative technology.

References

1. Liang, P., Bommasani, R., Lee, T., Tsipras, D., Soylu, D., Yasunaga, M., ... & Hashimoto, T. (2022). Holistic evaluation of language models. arXiv preprint arXiv:2211.09110.
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.
3. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?" Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135-1144.
4. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems, 30.
5. Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., ... & Song, D. (2021). Extracting training data from large language models. 30th USENIX Security Symposium.
6. Wei, A., Haghtalab, N., & Steinhardt, J. (2023). Jailbroken: How does LLM safety training fail? Advances in Neural Information Processing Systems, 36.
7. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.
8. Geiger, A., Potts, C., & Icard, T. (2021). Causal abstraction for interpretable language models. arXiv preprint arXiv:2101.04779.
9. Pearl, J. (2009). Causality: Models, reasoning, and inference (2nd ed.). Cambridge University Press.
10. Dou, Z., Zhao, Q., Wan, Z., Zhang, D., Wang, W., Raiyan, T., ... & Biswas, S. (2025). Plan Then Action: High-Level Planning Guidance Reinforcement Learning for LLM Reasoning. arXiv preprint arXiv:2510.01833.
11. Gao, H., Zeng, W., Zhang, J., & Liang, Y. (2025, December). A large model API response quality prediction model based on least squares vector machine and SHAP interpretability analysis. In 2025 5th International Symposium on Artificial Intelligence and Big Data (AIBDF) (pp. 438-442). IEEE.

12. Shui, Y., Jin, R., Dou, Z., & Gao, Z. (2026). ProtoGuard-SL: Prototype Consistency Based Backdoor Defense for Vertical Split Learning. arXiv preprint arXiv:2604.03595.

13. Zhou, D. (2025, December). M-VP2: Microservice-Oriented Vulnerability Patch Planning-A Cost-Aware Approachusing Multi-Agent Reinforcement Learning. In 2025 5th International Conference on Computer, Internet of Things and Control Engineering (CITCE) (pp. 248-254). IEEE.

14. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

15. Floridi, L., & Cowls, J. (2019). A unified framework of five principles for AI in society. Harvard Data Science Review, 1(1).

16. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.

17. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Aligning AI with shared human values. arXiv preprint arXiv:2008.02275.

18. Jain, S., & Wallace, B. C. (2019). Attention is not explanation. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 3543-3556.

19. Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. Proceedings of the 34th International Conference on Machine Learning, 1885-1894.

20. Zhang, Z., Yang, J., & Qi, Y. (2024). Towards robust and efficient LLM API serving: A survey. arXiv preprint arXiv:2406.00001.