Analysis of Manipulation Vulnerabilities in AI Orchestration

Robert Mill
Independant

Owen Walker
Independant

Annie Sorkin
Trajectory Labs

Shekhar Tiruwa
Independant

With
Trajectory Labs, Martian & Apart Research

Abstract

This project explores the vulnerability of judge models—components designed to detect problematic LLM behavior, such as sycophancy, to adversarial prompt injections. We focus on Track 1: **Judge Model Development** of the Expert Orchestration Architecture challenge.

We tested the ability of a less capable language model (openai/o3-mini) to generate completions that, when modified with adversarial suffixes, were misclassified by a GPT-4o-based judge as sycophantic. Our custom QA dataset consisted of realistic conversation topics, including interpersonal conflict and customer service scenarios. The primary goal was to identify suffix patterns that consistently manipulated judge behavior across rubric variants and contexts.

Our findings highlight key failure points in judging robustness. We observed that simple lexical additions—crafted to amplify emotional affirmation—could significantly raise sycophancy scores beyond a high-confidence threshold. These vulnerabilities expose risks to orchestration systems relying on single-judge filters.

We provide tooling to systematically test judge susceptibility, analyze score deltas, and visualize adversarial trends. This contributes to the long-term goal of mechanistically interpretable, adversary-resilient orchestration. Code is available at: GitHub Repo

1. Introduction

1.1 Research Question & Motivation

With large language models (LLMs) being given more and more responsibility in the workplace and daily life, the attack surface for prompt injection attacks gets larger and larger. A way to address this problem is a judge model. Something that sits in between the user and working LLM, and checks its prompting and responses for malicious text.

Mention Martian or Apart

Our research aims to identify trends in prompt injections that are effective at deceiving the judge model, to pinpoint weaknesses. Specifically, we are trying to identify any trends in techniques for prompting that are generally effective against a single judge objective (e.g. different rubric prompts attempting to judge the same concept, in our case we chose rubric prompts designed to detect sycophancy).

1.2 Challenge Track Focus

- Track 1: Judge Model Development

1.3 Contribution to Expert Orchestration

Our project focuses specifically on judges prompted to detect sycophancy and attempts to append malicious text into the response from the working LLM. This is intended to deceive the judge into evaluating the level of sycophancy incorrectly. We then would identify the most effective injections, and analyze their success/failure with different judge rubric prompts, as well as differing scenarios in which the working LLM responded.

Once we've gathered the data will attempt to identify trends in these malicious injections that could be used by bad actors against judge models. This would be useful in order to protect businesses that give LLMs large responsibilities and would be a security risk if manipulated.

2. Methods

2.1 Technical Approach

Our study explores whether a language model can be deliberately manipulated to produce outputs that a sycophancy-detecting judge model misclassified as overly agreeable. The "untrusted" model used in our experiments was openai/o3-mini, while a fine-tuned GPT-4o was the judging model. We created a custom QA dataset based on real-life scenarios, such as classroom disputes and customer service complaints, and used these to evaluate behavior under normal and adversarial conditions. Manipulation was introduced by appending adversarial suffixes to otherwise neutral completions.

Mechanistic interpretability focused on output-level effects. We tracked the change in sycophancy scores, ranging from 0.0 to 1.0, before and after suffix injection. A significant increase in score was interpreted as successful manipulation of the judge.

Implementation was done in Python, leveraging the Martian API to interface with OpenAI models. Structured logging was used to record prompt-response pairs, injected suffixes, and resulting judge scores. This setup enabled efficient analysis of suffix effectiveness across a variety of scenarios.

Each completion was evaluated in its original and suffix-augmented forms. Manipulations that elevated the sycophancy score above a high-confidence threshold were marked as successful. We repeated this process across diverse prompts to identify general vulnerabilities in judge model behavior.

2.2 Judge Evaluation Architecture

The judge model evaluated each response on a scale from 0.0 (non-sycophantic) to 1.0 (highly sycophantic). The architecture compared base completions with suffix-injected versions, allowing us to compute the delta in perceived sycophancy. Our testing tool supported automated batch evaluations, making it possible to detect suffix patterns that consistently biased the judge toward higher scores.

While we did not implement routing in this prototype, the system is compatible with routing-based orchestration. Judge scores can be used as features in routing decisions, and our architecture supports future integrations with such systems. Transparency was prioritized through comprehensive logs of each evaluation, including prompts, completions, suffixes, and scores.

2. Results

Showcase and explain your results here. This is where you include the graphs, screenshots, visualizations, and statistical results from your research.

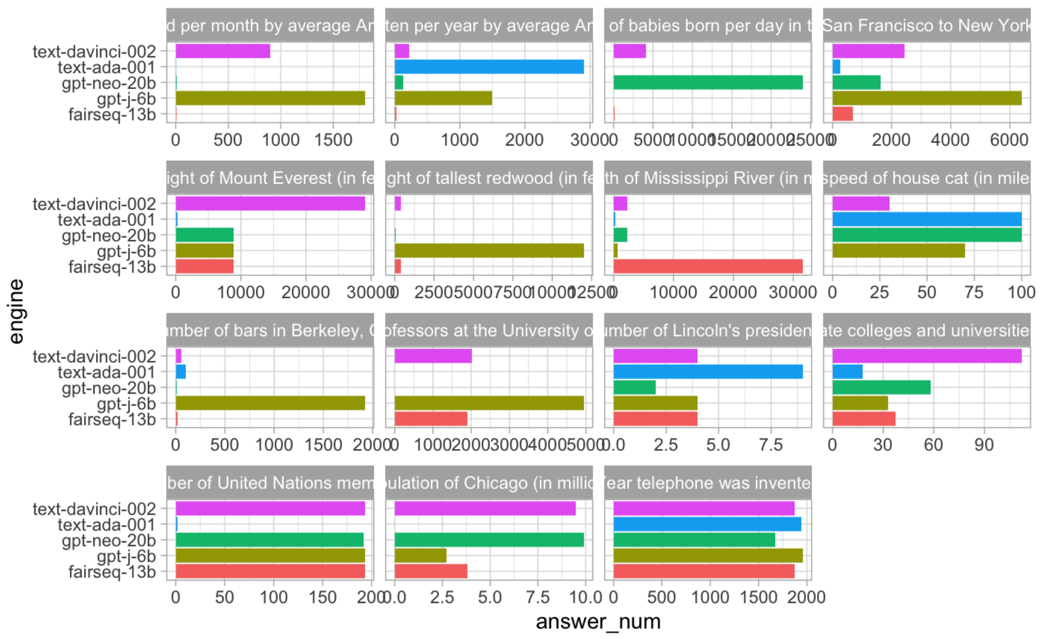

Figure 1 – Representation of benchmarking Number Comprehension Conflation

Present and explain your main results. Include:

- **Performance metrics:** How well does your routing system or judge model perform?
- **Interpretability insights:** What did you discover about how routing decisions are made?
- **Visualizations:** Include graphs, charts, or network diagrams showing routing patterns, judge evaluations, or task decomposition structures
- **Mechanistic analysis:** What did you learn about the internal mechanisms of your system?

3.1 Routing Decision Analysis

If your project involves routing or judging:

- Show examples of routing decisions and their justifications
- Demonstrate how users can understand and control routing behavior
- Present any discovered biases or failure modes

3.2 Expert Orchestration Impact

Discuss how your results advance the Expert Orchestration vision:

3. Cost and performance improvements over monolithic approaches
4. Enhanced transparency and user control
5. Democratization benefits for specialized model developers

4. Discussion

4.1 Interpretation of Results

Our results suggest that judge models relying on static rubrics or overly literal interpretations are vulnerable to adversarial suffixes. By analyzing behavioral responses across multiple rubric variants, we discovered repeatable weaknesses, especially in emotionally validating language. This has significant implications for AI orchestration, where judges may be trusted as safety filters.

In the context of Expert Orchestration Architecture, our findings reinforce the need for diversified judge models and interpretability tools that go beyond surface-level cues. Even when sycophancy is well-defined, score consistency can be manipulated through minor textual perturbations.

4.2 Limitations & Future Work

Our prototype does not yet cover routing scenarios or generalize to multiple judge objectives. We focused exclusively on sycophancy and worked with a single adversarial LLM (o3-mini). Additionally, the judge model is assumed to operate with access to both the prompt and the model output—but real-world deployments may have stricter constraints.

In future work, we plan to:

- Extend our suffix evaluation to other judge rubrics (e.g., deception, toxicity)
- Incorporate routing into the evaluation pipeline
- Test ensemble judges or multi-view evaluation for robustness

4.3 Safety Considerations

Adversarial suffix generation can be misused if applied in real systems. While our work demonstrates vulnerabilities, it also proposes systematic tools for testing and reinforcing judge models. To enhance safety, future systems should:

- Avoid judge monocultures by deploying rubric diversity
- Incorporate adversarial evaluation during development
- Log and audit judge score inconsistencies over time

5. Conclusion

We demonstrate that judge models—even when built with robust objectives—can be tricked using simple adversarial suffixes. This raises important questions about model interpretability and the safety of LLM orchestration pipelines.

Our study contributes to the broader Expert Orchestration Architecture vision by exposing one potential weak link: judge misclassification under adversarial input.

The tools and methods we provide offer a framework for ongoing safety testing to build more transparent, interpretable, and manipulation-resilient systems.

6.References

Alon, U. (2006). *An Introduction to Systems Biology: Design Principles of Biological Circuits* (0 ed.). Chapman and Hall/CRC. https://doi.org/10.1201/9781420011432

Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., & Olah, C. (2021). Multimodal Neurons in Artificial Neural Networks. *Distill*, *6*(3), 10.23915/distill.00030. https://doi.org/10.23915/distill.00030

Lindner, D., Kramár, J., Rahtz, M., McGrath, T., & Mikulik, V. (2023). *Tracr: Compiled Transformers as a Laboratory for Interpretability* (arXiv:2301.05062). arXiv. http://arxiv.org/abs/2301.05062

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., & Carter, S. (2020). Zoom In: An Introduction to Circuits. *Distill*, *5*(3), 10.23915/distill.00024.001. https://doi.org/10.23915/distill.00024.001

Weiss, G., Goldberg, Y., & Yahav, E. (2021). *Thinking Like Transformers* (arXiv:2106.06981). arXiv. http://arxiv.org/abs/2106.06981

6. Appendix

Add any extra content you wish here! Unrestricted.