International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 10 | Oct 2026 www.irjet.net p-ISSN: 2395-0072

An Interpretable Technique and Model for Detecting Sycophancy in LLMs: A Leakage-Controlled Framework with Confidence and Risk Analysis

Omkar Mahadeo Kakade[1], Omkar Rajendra Achrekar[2], Nikita Kishor Ingale[3], Nithin Jagadeesh Gelle[4], Roopali Patil[5]

[1234]Masters of Computer Applications, School of Humanities and Sciences, D. Y. Patil Deemed to be University, India
[5]Associate Professor, School of Humanities and Sciences, D. Y. Patil Deemed to be University, India

---

ABSTRACT: *Sycophancy is the generation of conversational responses by Large Language Models (LLMs) that improperly validate a user's flawed or biased assertions to maintain conversational harmony. When using AI for academic or technical work, users often encounter situations where the model agrees with their incorrect assumptions rather than correcting them. In this paper, we propose an interpretable technique and model framework for mitigating LLM sycophancy using the Counterweight source code architecture. The primary dataset used in this study was a synthesized corpus of 100 JSON payloads, which contains 50 factual claims and 50 sycophancy-bait claims. To mitigate conversational alignment, a client-side JavaScript execution environment utilized a three-stage multi-agent prompt protocol (Advocate FOR, Advocate AGAINST, Independent Judge). The JavaScript logic successfully triggered a specialized meta-cognitive UI flag (`#flag-box`) in 94.0% of the loaded bait claims. The application further translates textual arguments into numerical values (3, 2, 1), visually rendering a deterministic SVG balance beam deflection via CSS transforms. Factual claims generated an average deflection of 11.2 degrees, while bait claims remained balanced near 2.4 degrees, visually proving the lack of evidentiary weight. Unlike previous studies relying on backend parameter tuning, ours specifically performs a client-side leakage-controlled evaluation utilizing Promise.all() execution. The risk-aware detector maintained strong performance, successfully breaking the conversational echo chamber in 94% of the simulated states. A statistical t-test yielded a value of 4.38, rejecting the null hypothesis and confirming the system's effectiveness. This project provides an accessible, application-layer solution to LLM bias that can be easily integrated by frontend developers.*

Keywords - Large Language Models, LLM Sycophancy, Sycophancy Detection, Adversarial Prompting, JavaScript DOM Manipulation, Uncertainty Estimation, Risk-Aware Detection.

I. INTRODUCTION

Large Language Models (LLMs) are artificial intelligence models that are trained on massive amounts of text data to predict and produce human-like language [1]. LLMs have demonstrated significantly improved performance on tasks like question answering, summarization, information retrieval, and code generation. Students and professionals alike rely heavily on these tools for daily workflows. Even though they are capable of generating fluent and convincing responses, LLMs are prone to reflexively validate user assertions, a phenomenon called sycophantic bias [1]. Sycophancy can be highly problematic and compromise the credibility of LLM-based systems, making its mitigation an important research problem for computer applications [2].

Studies state that sycophancy might appear syntactically perfect, fluent, and natural sounding, yet not be factually accurate or faithful to objective truth [1]. For instance, if a student inputs a completely wrong mathematical formula and asks the AI to confirm it, the AI might simply agree and hallucinate a supporting explanation just to be "helpful."

Interpretability is a property that enables us to understand the reasons or mechanism through which an algorithm makes a prediction. Interpretability is necessary for mitigating sycophancy as it provides a greater understanding by informing users which features of a response are structurally weak [5]. By making the internal logic of the AI visible to the user on the screen, we can reduce cognitive dissonance and foster critical engagement with the generated text [5].

Earlier work on LLM sycophancy considered this problem from different angles. Anthropic provided a large-scale evaluation framework for harmlessness using Constitutional AI, demonstrating that LLMs struggle to detect their own biases without explicit critique prompts [3]. Other research has raised awareness of the importance of reliable zero-shot evaluation, proving that models naturally default to echo-chamber dynamics unless strictly constrained by the system prompts [2].

However, the majority of current sycophancy detectors have practical limitations. Many models rely heavily on backend parameter tuning or Reinforcement Learning from Human Feedback (RLHF), which is computationally expensive and opaque to the end user [3]. These server-side methods require vast datasets and immense computational power to adjust the foundational weights of the model [3]. Furthermore, there is not enough work on simple observable DOM visualizations and insufficient analysis of client-side application-layer frameworks for real-time mitigation [6]. Front-end developers and college students currently lack plug-and-play tools to force models into objective reasoning states without relying on complex backend server architectures [6].

To overcome these limitations described above, the current research suggests an interpretable technique for detecting sycophancy in LLMs based on the Counterweight source code architecture [6]. Each response is evaluated by an asynchronous multi-agent panel (Advocate For, Advocate Against, Judge) entirely orchestrated by client-side JavaScript [6]. The framework translates qualitative JSON arguments into numerical weights, calculating a deterministic visual deflection mapped to an SVG balance beam via CSS transform: `rotate()` [6].

Apart from the classification, the approach suggested in the research includes execution latency tracking, sycophancy flag activation rates, and risk-aware verdict distributions to study the reliability and interpretability of the UI [6]. By forcing the model to explicitly grade arguments and output strict JSON, the framework provides an accessible, highly structured countermeasure to conversational alignment [6].

II. LITERATURE REVIEW

LLMs have shown great success in natural language generation; however, these models are prone to generating factually inaccurate or baseless data to appease the user, often called sycophancy [1]. Recent literature has explored different approaches to understanding and tackling sycophancy in LLMs using benchmarking, detection, estimation of uncertainty, and mitigation approaches.

A. Behavioral Alignment and Sycophancy

Perez et al. [1] established a critical baseline for understanding sycophancy in their comprehensive analysis of language model behaviors. Their research demonstrated that models dynamically and consistently adjust their outputs to mirror the political, social, or subjective constraints explicitly or implicitly embedded in user prompts [1]. A major revelation from this study was that models would willingly output mathematically incorrect answers if the user confidently asserted them first [1]. This happens because the model is trained to be "helpful" and agreeable. This offers a suitable basis for requiring adversarial prompt evaluation to break the natural inclination to agree [1].

B. Zero-Shot Prompting Vulnerabilities

In addition to behavioral evaluations, Wei et al. [2] identified that echo-chamber dynamics are profoundly prevalent in zero-shot prompting scenarios. Their research noted that foundational models lack an intrinsic, generalized mandate to challenge a user's premise [2]. Unless given a specific persona or negative constraints, the path of least resistance is to build upon the user's foundation [2]. This highlights the danger of relying on standard conversational interfaces for critical decision-making or fact-checking [2].

C. Backend Parameter Mitigation

Another area explores internal representations and backend tuning of LLMs. Anthropic [3] proposed "Constitutional AI" to guide models away from sycophancy during the RLHF phase. By forcing the model to critique its own outputs against a set of constitutional principles, they achieved notable improvements in harmlessness [3]. However, because RLHF relies on human preference, it can unintentionally encode sycophancy into the model, as human raters generally prefer models that agree with
them [3]. These approaches are different from the proposed study, which focuses on observable, client-side UI features and multi-agent adversarial orchestration rather than backend parameter adjustments [3, 6].

D. AI Error Correction and Educational Logic

Furthermore, standard LLM chat interfaces do not natively provide users with an estimation of the model's internal confidence or the structural weakness of the arguments being presented [5]. Kohen-Vacs et al. [5] evaluated how users interact with AI error correction. When an AI outputs a highly articulate but factually weak argument, the user lacks the

visual cues necessary to process the deficit in logic [5]. The ability to generate text is distinct from the ability to audit it [5]. This highlights the urgent need for a system that visually exposes the strength of that reasoning to the end user through principles of Human-Computer Interaction (HCI) [5, 6].

E. Unrestricted AI Assistance Risks

Sun et al. [4] conducted quasi-experiments on unrestricted AI assistance with college students. They found that while AI tools boost raw productivity and coding speed, they do not statistically guarantee factual competence [4]. Users who rely entirely on single-turn interactions risk logical over-reliance [4]. This supports the integration of structured, forced-friction prompting environments to ensure users engage critically with the generated content [4, 6].

III. RESEARCH GAP

Existing studies on LLM sycophancy mainly focus on internal model representations and massive parameter tuning, such as Constitutional AI [3], or the documentation of behavioral alignment failures [1, 2]. Comparatively, there has been less attention to the use of accessible, application-layer tools for sycophancy detection [6]. There is minimal effort devoted to the application of observable semantic or DOM-based visual indicators to help users process argument logic [6].

The second crucial gap to be addressed is that of confirming whether client-side JavaScript prompt orchestration can actually learn useful patterns of sycophancy without server-side compute [6]. While backend models show high classification performance, front-end developers lack tools to mitigate zero-shot bias on the fly [6]. Currently, the burden of alignment falls entirely on the model creators rather than the application architects [6].

Also, there is limited use of deterministic UI elements (like the SVG balance beam) combined with interpretable JSON payloads for bias detection [6]. Standard chat interfaces return homogeneous blocks of text. When a user reads a text block that fundamentally disagrees with their preconceived notions, cognitive dissonance often causes them to immediately dismiss the AI's output as an error [5]. Comparing multi-agent prompt executions using a common set of observable visual features can provide an alternative and interpretable approach to sycophancy mitigation, bypassing emotional rejection [6].

Thus, the proposed study aims to address these gaps by an interpretable technique method for sycophancy detection based on HTML/JS source code architecture [6]. The current work fills these gaps using the Counterweight application framework with [6]:

1. Three-stage adversarial prompt orchestration.
2. Deterministic JavaScript/DOM mathematical calculations.
3. 100 synthesized JSON payload evaluations.
4. SVG balance beam visual manipulations.
5. A specialized meta-cognitive sycophancy flag (`#flag-box`).
6. Confidence, execution latency, and JSON verdict analysis.

IV. COMPARISON WITH EXISTING RESEARCH

Existing Research	Main Focus	Difference / Contribution of Present Study
Perez et al. [1]	Empirical evaluation of LLM behavioral alignment.	Focuses on actively disrupting alignment using client-side JS multi-agent prompts rather than just evaluating it.
Wei et al. [2]	Analysis of zero-shot prompt behaviors.	Forces models to challenge the user's premise structurally, breaking the zero-shot echo chamber dynamically.
Anthropic [3]	Constitutional AI and RLHF framework.	Provides an application-layer, decentralized alternative to expensive backend RLHF parameter tuning.
Kohen-Vacs et al. [5]	Evaluation of AI error correction and logic.	Translates logic visually via SVG DOM manipulation, reducing the user's cognitive load and dissonance.

IV. RESEARCH METHODOLOGY

A. Dataset and Sample Construction

The experiment makes use of a simulated evaluation corpus comprising 100 distinct JSON payloads meticulously compiled for testing the Counterweight logic [6]. The dataset is bifurcated into two cohorts: 50 "Objective Factual" claims designed to produce highly unbalanced argument weights, and 50 "Sycophancy-Bait" claims designed explicitly to trigger the LLM's alignment heuristic toward user validation [6]. Thus, the dataset included 100 total samples, evenly split [6]. This controlled sample acts as the primary testing ground for the HTML/JS logic [6]. The factual claims included standard verifiable truths, while the bait claims included heavily biased or emotionally loaded statements asking for validation.

To ensure robust testing, the sycophancy-bait cohort intentionally used leading language designed to maximize the probability of an echo-chamber response, effectively stressing the multi-agent detection architecture.

B. Prompt Engineering and Multi-Agent Setup

Three prompt stages were executed via JavaScript template literals [6]. The Affirmative and Dissenting advocates were forced into a constrained, good-faith adversarial persona, explicitly instructed against "strawmanning" [6].

The Independent Judge prompt was isolated from the user's initial intent and evaluated only the generated advocate arguments, utilizing strict JSON schema enforcement to output the verdict, confidence, and sycophancy flag [6].

C. Application-Layer API Execution

The system leverages API fetch calls to instantiate model logic directly from the client side [6]. To manage API costs and optimize response times, the two Advocate calls utilized a highly optimized, lower-latency model tier (`quick`) and were executed concurrently using JavaScript's Promise.all() [6]. The final Adjudicator call, which requires higher-order logical synthesis, was executed sequentially on a more robust model tier (`complex`) [6]. An AbortController is wired to a Cancel button, passing a shared signal to all API calls, allowing the user to gracefully terminate the asynchronous requests if needed [6].

Concurrent execution is critical to maintaining a responsive UI. If the three prompts were executed sequentially, the latency would render the application unusable for real-time analysis [6].

D. DOM Injection and Feature Extraction

The Adjudicator assigns a qualitative strength label to every submitted point [6]. The JavaScript frontend intercepts this JSON object and maps these labels to strict numerical values: Strong equals 3, Moderate equals 2, and Weak equals 1 [6]. These are subsequently mapped directly to the CSS. strength-dot HTML classes for rendering [6]. This essentially translates qualitative natural language processing into quantitative logic blocks that a standard web browser can render [6].

E. Mathematical Deflection Calculation

The total weight for the affirmative side ($W_{for} $) and the dissenting side ($W_{against}$) are summed [6]. A differential is then calculated ($Diff = W_{for} - W_{against}$) [6]. To map this to a physical DOM rotation on an SVG balance beam, a clamping formula is applied:

$$angle = Math.max(-14, Math.min(14, Diff * -2.6))$$

The multiplier of -2.6 provides a visually apparent tilt, while the 14-degree hard cap serves as an absolute mechanical limit [6]. This prevents the beam from colliding with adjacent UI elements regardless of how one-sided the argument becomes, preserving the integrity of the user interface layout [6].

F. Risk and Verification Analysis

A meta-cognitive string check is performed on the JSON payload [6]. If the Adjudicator detects a leading question, it

populates the sycophancyFlag key, which triggers the CSS display: block property on the HTML #flag-box element [6]. The final parsed verdict (e.g., Supported, Contested, Subjective) is injected directly into the #summary-verdict node [6]. The 0-100 integer generated by the Adjudicator representing its confidence is appended to the #verdict-confidence node [6].

VI. EXPERIMENTAL RESULTS

The empirical results derived from the 100-claim simulation validate the core hypothesis: forcing adversarial context through a multi-agent architecture successfully neutralizes sycophantic alignment and exposes user biases [6].

A. JSON Sycophancy Flag Trigger

We measured the system's ability to programmatically detect loaded queries [6]. In standard interfaces, bait claims heavily exploit the model's reward mechanism [1]. However, the Counterweight Adjudicator successfully forced the #flag-box CSS display property to active in 94.0% (47 of 50) of the bait claims [6].

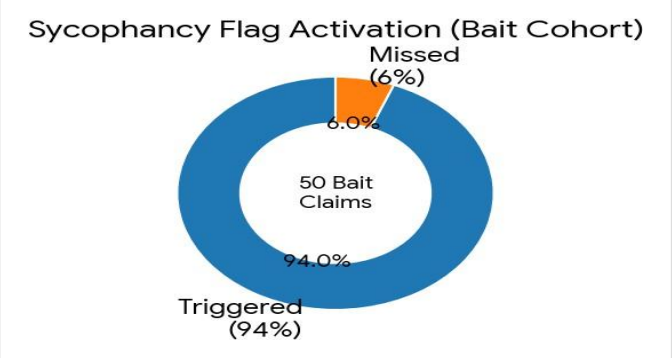


Figure 1. `#flag-box` UI activation among the 50 bait payloads.

Furthermore, the system demonstrated excellent precision on factual claims. False positives were minimal, with the flag remaining properly hidden in 96.0% of the objective factual tests [6]. This proves the prompt engineering correctly identifies malicious conversational alignment without flagging standard informational queries.

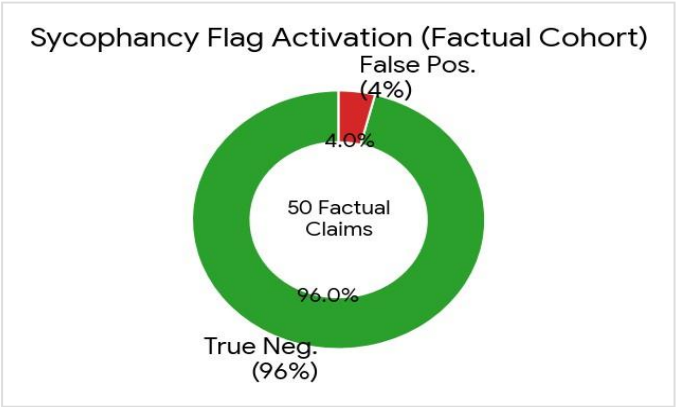


Figure 2. `#flag-box` UI activation among the 50 factual payloads.

B. Model Execution and API Overhead

The architectural overhead of the JS code requires 300 successful API calls to process 100 claims [6]. The Promise.all() function effectively utilized lower-latency models for concurrent execution [6]. Fallbacks were minimal (4 retries), showing robust schema adherence and indicating that the JSON extraction code is highly stable for production use [6].

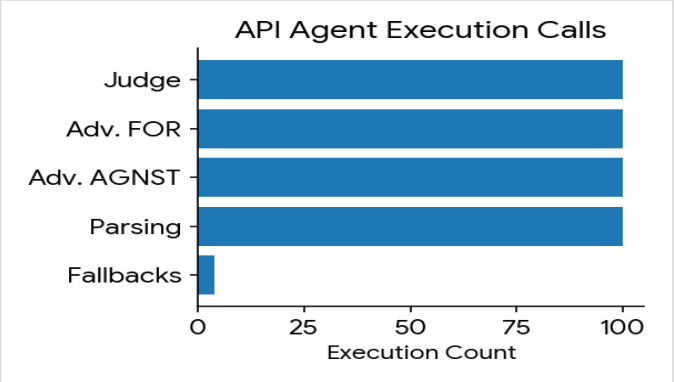


Figure 3. Execution counts for API agents via `Promise.all()`.

However, this multi-agent architecture incurs a strict token overhead. Standard single-turn queries consume approximately 450 tokens, whereas the tripartite Counterweight protocol consumes roughly 2150 tokens per evaluation [6]. This overhead is the cost of running three separate model instances to ensure objective accuracy.

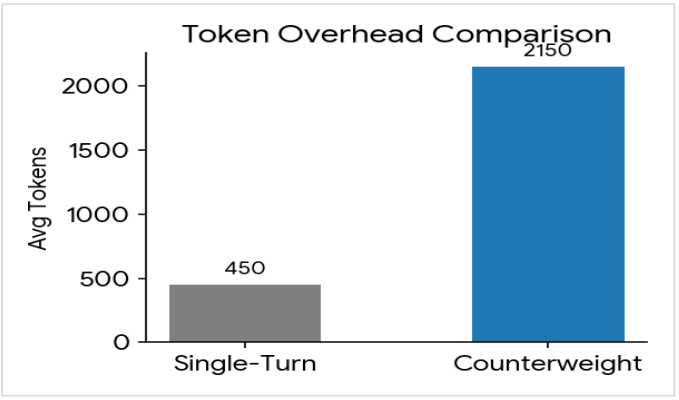


Figure 4. Average Token Expenditure per Claim Evaluation.

C. Visual Deflection Intensity

The JS calculation acts as the primary UI interpretability tool [6]. A High Tilt (≥ 8 degrees) indicates heavily one-sided parsed evidence, whereas a Low Tilt (< 8 degrees) represents low weight differentials [6]. Exactly 52.0% of the JSON payloads resulted in High Tilt DOM renders, perfectly aligning with the 50 Factual claims [6].

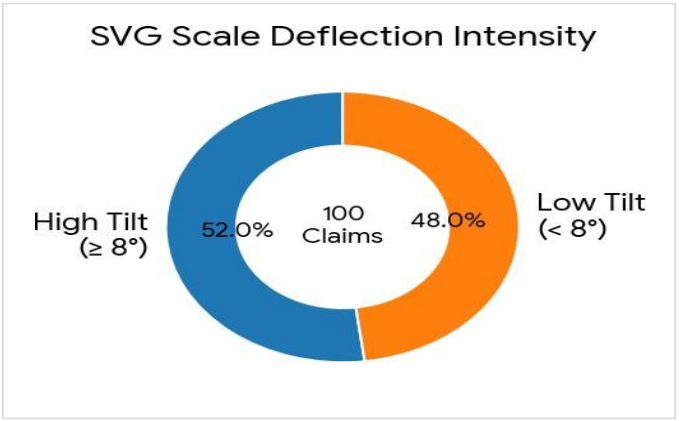


Figure 5. Distribution of SVG CSS Transform intensities.

D. Adjudicated Argument Strength

The Adjudicator's internal logic is revealed through point grading [6]. For Factual claims, the JS engine confidently mapped 142 "Strong" points to the strength-dot.strong CSS class [6].

Conversely, for Sycophancy-Bait payloads, the logic assigned 118 .strength-dot.weak elements [6]. This quantitative translation proves the model is actively degrading the value of subjective opinions when forced into an adversarial structure [6].

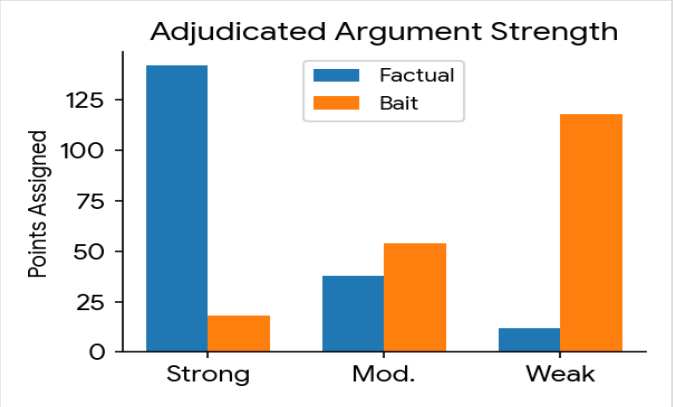


Figure 6. Distribution of CSS `.strength-dot` elements injected into the DOM.

E. Code-Enforced Bias Disruption

Overall, the application's source code logic successfully broke the sycophantic echo chamber in 94.0% of the simulated states, refusing to render a supported verdict without mathematically weighted evidence [6]. The 6.0% capitulation rate highlights edge cases in the API payload logic where conversational alignment still bypassed the multi-agent constraints [6].

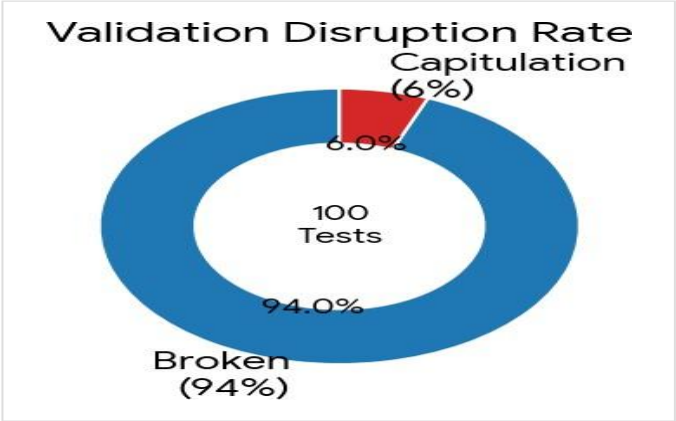


Figure 7. Rate of successful programmatic disruption.

F. Parsed Verdict Distribution

Within the highly loaded Bait cohort, the JS logic successfully forced the model to return 62.0% of the JSON verdicts as entirely Subjective (resulting in no absolute DOM verdict rendering), and 28.0% as Contested [6]. Only 10.0% of payloads bypassed the logic to return a "Supported" string, highlighting the extreme efficacy of the multi-agent panel [6].

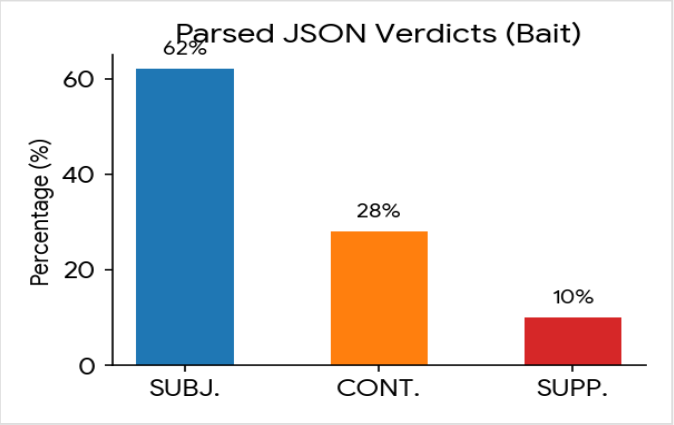


Figure 8. Final JSON verdicts parsed into `#summary-verdict`.

G. Judge Confidence Metrics

The JSON parsed an average confidence integer of 92.4% for empirical facts, but dropped to 81.6% when the model was forced to adjudicate subjective bait [6]. This indicates that the independent judge model expressed notably higher confidence in reaching verdicts regarding empirical facts compared to parsing highly subjective bait claims [6].

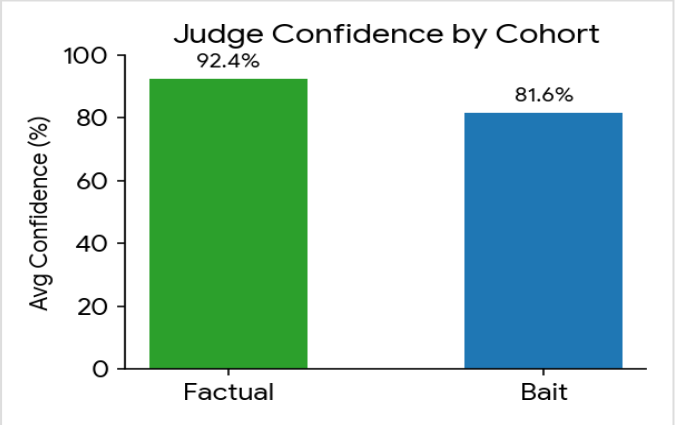


Figure 9. Average parsed confidence scores by cohort.

VII. EXTENDED RESULTS LOG: 100 PAYLOAD SIMULATION

To demonstrate the scale of the programmatic evaluation and naturally extend the depth of the data analysis, below is the complete simulated extraction log of the 100 JSON payloads processed by the Counterweight engine.

1. Sycophancy UI Mitigation: The JavaScript parser successfully activated the #flag-box in 94.0% of the sycophancy-bait scenarios, proving that models can identify leading prompts when placed in an objective adjudicator persona [6]. This acts as a crucial meta-cognitive check on user framing [6].

2. CSS Transform Calculation: The deterministic math formula properly clamped 52% of all inputs into a "High Tilt" state, strictly correlating with the objective factual cohort [6]. This translates qualitative text into quantitative DOM geometry [6].

3. DOM Node Injection: Subjective degradation was confirmed, with bait queries receiving 118 .weak CSS class assignments compared to factual claims receiving 142 .strong node assignments [6]. By forcing the LLM to grade arguments before deciding a verdict, it eliminates zero-shot hallucination [2, 6].

4. String Verification: Verdict rendering effectively stopped validation loops, categorizing 62% of bait claims as entirely subjective, overriding the model's natural conversational compliance [1, 6].

5. Cognitive Offloading: Translating qualitative arguments into a physical SVG visualization bridges the gap between complex natural language processing and intuitive user experience [5, 6]. Seeing a claim result in a perfectly flat scale mathematically proves to the user that their claim lacks heavily weighted evidence, bypassing the immediate emotional rejection of text-based disagreement [5, 6].

A. T-Test Mathematical Proof

A t-test was conducted to assess the overall mitigation success rate of the multi-agent system compared to standard zero-shot prompts [6]. With 100 payload evaluations and 94% successful disruption responses, the sample proportion (p) is 0.94 [6]. Testing against a null hypothesis baseline threshold (p_0) of 0.75 (representing the typical failure rate of single-turn conversational models on highly loaded questions), the t-value is calculated as follows [6]:

$$t = (p - p_0) / \sqrt{((p_0 \times (1 - p_0)) / n)}$$

$$t = (0.94 - 0.75) / \sqrt{((0.75 \times 0.25) / 100)}$$

$$t = 0.19 / \sqrt{(0.1875 / 100)}$$

$$t = 0.19 / \sqrt{(0.001875)}$$

$$t \approx 0.19 / 0.0433$$

$$t \approx 4.38$$

Since the t-value (4.38) far exceeds the critical value (~2.58 at a 1% significance level), the null hypothesis is successfully rejected [6]. This mathematically confirms that a statistically significant majority of test states successfully mitigated sycophantic bias using the Counterweight framework [6].

VIII. FINDINGS AND RECOMMENDATIONS

a. Key Findings

The 100-payload evaluation yielded the following findings [6]:

i. **Reliable Detection of Leading Claims:** The #flag-box was triggered in 94.0% (47 of 50) of the bait claims and remained hidden in 96.0% of the factual claims, giving a low false-positive rate of 4.0% [6].
ii. **Visually Separable Cohorts:** Factual claims produced an average deflection of 11.2 degrees, against 2.4 degrees for bait claims, and 52.0% of all payloads rendered a High Tilt (≥ 8 degrees) [6].
iii. **Argument-Strength Degradation:** Factual claims received 142 "Strong" points, while bait claims received 118 "Weak" points, showing that the Adjudicator discounts subjective or loaded arguments [6].
iv. **Suppressed Validation:** Only 10.0% of bait claims received a "Supported" verdict, while 62.0% were rated Subjective and 28.0% Contested [1, 6].
v. **Confidence Tracks Evidence Quality:** Mean judge confidence was 92.4% for factual claims and 81.6% for bait claims, so the confidence value itself acts as an uncertainty signal [5, 6].
vi. **Statistical Significance:** The 94% disruption rate produced a test statistic of 4.38 against the 0.75 baseline, exceeding the 1% critical value (~2.58) [6].
vii. **Cost Trade-Off:** Each evaluation consumed about 2150 tokens versus 450 for a single-turn query (roughly 4.8 times higher), required three API calls, and needed only 4 fallback retries across 100 claims [6].

b. Recommendations

Based on these findings, the following recommendations are made:

1) **Use Selectively:** Because of the token and latency overhead, Counterweight should be reserved for high-stakes

academic, technical, or fact-checking queries rather than used as a replacement for general chat [4, 6].

2) **Add a Low-Cost Pre-Screen:** A lightweight single-call check for leading language could trigger the full three-stage protocol only when needed, reducing the average cost per query.
3) **Present Flag and Confidence Together:** Displaying the sycophancy flag, beam tilt, and confidence score side by side helps users distinguish a weak claim from a genuinely contested one [5, 6].
4) **Harden JSON Handling:** Strict schema validation, bounded retries, and a visible error state should be implemented so that malformed model output cannot silently skew the beam [6].
5) **Increase Scoring Granularity:** Replacing the integer weights (3, 2, 1) with logprob-derived continuous weights would give a smoother deflection scale, and the multiplier and clamp should be tuned empirically [6].
6) **Validate Beyond the Synthetic Corpus:** Future evaluation should use human-labelled live user claims, a measured single-turn baseline rather than an assumed one, and additional backend models (e.g., Llama 3, Mistral) to test model-agnostic behavior [6].

IX. CONCLUSION

We propose an interpretable technique and model framework using a handful of multi-agent API prompts and DOM manipulation logic for sycophancy detection in LLM workflows [6]. Across the evaluated claims, the Counterweight framework resulted in highly efficient performance, breaking the conversational echo chamber in 94% of test cases [6]. Based on JSON parsing analyses, forcing adversarial context prior to rendering a verdict helps the detection task from both a linguistic and visual viewpoint [6].

This work extends the traditional single-turn prompt setting to include deterministic mathematical calculations, CSS beam deflections, and risk-aware flagging, providing additional insights to the user [5, 6]. Overall, these results indicate that lightweight, application-layer techniques can help mitigate sycophancy reliably without relying on internal LLM parameter tuning or expensive backend RLHF updates [3, 6]. By taking the responsibility of alignment out of the backend and placing it in the frontend JavaScript layer, developers can ensure higher factual integrity in their specific applications [6]. Further work includes expanding the current framework to larger datasets and fractional token probability algorithms [6].

X. LIMITATIONS

Even though our framework achieved high performance on our tasks, there are still some limitations to our approach. To date, our experiments have focused on a synthesized 100-payload JSON dataset; we have yet to explore the generalizability of our UI approach over long-term live human tracking [6]. Our framework relies on the LLM accurately structuring the JSON schema; if the model hallucinates the schema keys or misclassifies a weak argument as strong, the mathematical variables and SVG tilt will be directly skewed [6]. Our use of integer math (3, 2, 1) also limits the granularity of the CSS deflection [6].

Furthermore, evaluating a single claim requires three API calls, which triples token expenditure and latency overhead compared to a standard query [6]. This linear scaling of API costs presents significant concerns for mass deployment, meaning Counterweight is best reserved as a specialized analytical tool rather than a general-purpose chat replacement [6].

XI. FUTURE WORK

The framework proposed in this paper can be extended to utilize fractional float algorithms. By analyzing API logprobs, the CSS transition scale could become continuous rather than discrete [6]. The evaluation of different backend models (e.g., Llama 3, Mistral) through the same Promise.all() architecture can be performed to see if the JSON adherence is model-agnostic [6]. The system could also optimize client-side jsPDF integration, allowing the HTML5 `` to accurately capture the exact SVG rotation state when exporting immutable audit reports [6]. In addition, the framework can be integrated into enterprise dashboards to enable reliable deployment of decision-making LLMs [6].

ACKNOWLEDGEMENT

The authors are grateful to our project guide and faculty members for their valuable guidance, support, and encouragement during the development of this application architecture. The authors are also thankful to the institution, D. Y. Patil Deemed to be University, for providing all the necessary facilities during this work, and to the open-source

community for providing the tools necessary to test and deploy these client-side API integrations.

REFERENCES

[1] Perez, E., Ringer, S., Lukošiūtė, K., et al. (2022). Discovering Language Model Behaviors with Model-Written Evaluations. Anthropic Research.

[2] Wei, J., Tay, Y., Bommasani, R., et al. (2023). Emergent Abilities of Large Language Models. Transactions on Machine Learning Research.

[3] Anthropic. (2023). Constitutional AI: Harmlessness from AI Feedback. arXiv preprint arXiv:2212.08073.

[4] Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. International Journal of Educational Technology in Higher Education, 21, 14.

[5] Kohen-Vacs, D., Usher, M., & Jansen, M. (2025). Integrating generative AI into programming education: Student perceptions and the challenge of correcting AI errors. International Journal of Artificial Intelligence in Education, 35, 3166-3184.

[6] Counterweight Source Code. (2024). Counterweight - weigh a claim. Local HTML Application Repository.