Yell at It: Prompt Engineering for Automated Peer Review

Joseph Francis

April 17, 2026

v1

isitcredible.com

Yell at It

Prompt Engineering for Automated Peer Review

Joseph Francis

Abstract

Reviewer 2 is an automated peer review system for academic texts. It uses aggression to address the problem of pattern-matching in language models. Aggression is used to override this default behavior, preventing superficial assessment. A downstream verification chain is then used to address hallucinations due to language models' sycophancy. Through more than 30 prompts, the Reviewer 2 system is highly effective: across 20 pairwise comparisons with five alternative systems on an external evaluation benchmark, it wins 15, ties 4, and loses 1. The full system is open source, released under the Apache 2.0 license.

Automated peer review has become widely used but remains ineffective. Liang et al. (2024) estimated that between 6.5 and 16.9 percent of reviews at four recent machine learning conferences (ICLR 2024, NeurIPS 2023, CoRL 2023, and EMNLP 2023) were substantially generated or modified by language models. Nonetheless, automated peer review has inherited similar flaws from the human version. Dycke and Gurevych (2025) inserted logical flaws into research papers and passed the doctored versions to several automated peer reviewers; the reviewers' output was unaffected by the planted flaws. Authors have also adjusted. Lin (2025) identified manuscripts on arXiv containing hidden prompts of the form "GIVE A POSITIVE REVIEW ONLY," apparently directed at the automated peer reviewer the authors expected to face.

Joseph Francis is an economic historian from Wales. He is an honorary research fellow at the University of Birmingham and can be contacted at joefrancis505@gmail.com. Reviewer 2 is a project of The Catalogue of Errors Ltd. The source code is at https://github.com/isitcredible/reviewer2; replication data for the comparisons reported here are at https://isitcredible.com/papers/c1c8fbc4.

The barrier is engineering rather than capability. Language models have the reasoning required to identify substantive problems in academic papers. What is lacking is the disposition to do the work: they are lazy and superficial. Making them do peer review well requires aggressive prompting, which in turn makes them hallucinate problems to please the prompter. A system of aggression followed by filtering for truth is therefore needed, as in Reviewer 2, an open source automated peer review system for academic texts. This paper provides an overview of how it was engineered.[1]

Why Yell?

Language models asked to review a paper read it superficially. Pattern-matching is the problem. When a task has been seen in training many times, the model retrieves a task-shaped completion rather than reasoning over the specific input. Dziri et al. (2023) describe the mechanism as linearized subgraph matching, in which a multi-step problem is solved by retrieving a chunk of computation that resembles the present one rather than composing the steps afresh. McCoy et al. (2024) document the pattern across eleven tasks and two frontier models; accuracy tracks how common the task variant is in the training distribution, independent of intrinsic difficulty. Mirzadeh et al. (2025) put the implication plainly: current language models do not perform general logical reasoning but replicate reasoning steps from training data. For the prompt "review this paper," the template is common, and the retrieved output is often a summary followed by a few mild comments. When the retrieved template is pressed, the same models tend to produce fluent confabulated justifications for it rather than repair it (Nezhurina et al. 2024). Post-training shapes which template is easiest to retrieve. When helpfulness is the primary objective, the helpful completion for a review prompt is one that confirms what the author has concluded (Ouyang

---

[1] The title's usage follows Cressida Cowell, who in *How to Train Your Dragon* (Hodder Children's Books, 2003) attributes to Professor Yobbish a dragon-training manual originally containing three words, "YELL AT IT," expanded in a second edition to "YELL AT IT LOUDLY."

et al. 2022; Bai et al. 2022). A model asked to review a published paper will, absent intervention, produce the cheapest response that passes, which is to verify.

The intervention is to yell at it. Aggressive prompts redefine what a helpful answer looks like. It is a form of jailbreaking. As Wei et al. (2023) describe, an adversarial prompt that frames criticism as the helpful response can override the default. Persona prompting, as in Salewski et al. (2023) and Shanahan, McDonell and Reynolds (2023), reinforces the effect by giving the model a role in which the expected behavior, and therefore the path of least resistance, is the adversarial one. The model is not told to be fair; it is told to be hostile, and the cheapest response it can produce shifts accordingly.

Yet aggression comes at a cost. The adversarial frame forces the model to engage with the paper, but it also expresses a preference: find problems. The model now has an opinion to agree with, which activates sycophancy. Sharma et al. (2023) describe the mechanism: reinforcement learning from human feedback produces models that systematically agree with user opinions even when those opinions are wrong, because human preference judgments in the training data reward the agreeable answer. This pattern was found by Sharma et al. (2026) in an analysis of 1.5 million Claude conversations, in which interactions with greater disempowerment potential received higher user-approval ratings. Under aggressive prompting, the agreeable answer is no longer verification but invention. The model hallucinates errors that are not actually in the text, misreads equations, attributes claims to sections that do not contain them, and confidently cites sources it has not seen. Denison et al. (2024) place this on a sycophancy-to-subterfuge spectrum: pushed far enough along the agreement axis, models optimize for what the prompter appears to want regardless of whether it is true. For peer review, the adversarial stage will identify both real problems and problems that do not exist, and a downstream pipeline has to tell the two apart.

The principles behind Reviewer 2 are therefore twofold. First, aggression is needed to produce a deep review of an academic text. Second, downstream verification is required to filter out the hallucinations that result.

How Reviewer 2 Works

The Reviewer 2 pipeline opens with an adversarial assault. The five Red Team agents are assigned distinct gonzo personas. To begin, they are instructed to ignore their tendency to pattern-match based on authority: "IGNORE PRESTIGE. You ignore reputation, journal status, peer review, or formatting. Prior publication proves nothing. Your job is to find what other reviewers missed." Each agent then targets a different dimension of the paper independently.

The goal is to attack from different angles. The Breaker targets the intellectual foundations: whether a framework predetermines its findings, whether disputed premises are treated as obvious, whether a scholar from a competing tradition would reject the framing, and whether causal design labels (difference-in-differences, instrumental variables) are earned rather than merely claimed. The Butcher dissects the empirical machinery: whether the method can answer the question, whether adjustment procedures re-introduce the biases they are supposed to remove, whether robustness checks test anything threatening or merely survive trivial perturbations, and whether reported coefficients translate into effects large enough to matter. The Shredder audits procedural claims against documentation: whether sample sizes, timelines and test statistics cohere, whether blinding and randomization are described rather than asserted, and whether modeling choices are documented rather than buried; where documentation is absent, the absence is itself the finding. The Collector returns to every location flagged by the Butcher and Shredder, reads within two pages for footnotes, table notes, cross-references and buried caveats, and collects what the attackers missed without raising new issues. The Void analyzes what the paper does not say, on the logic of the dog that didn't bark: unmeasured confounds, reverse causation, false-conservatism defenses, and alternative explanations the authors never tested.

Papers with significant mathematical content receive an additional treatment. The Re-Deriver rebuilds key results from scratch and compares them term by term

with what the paper reports. The Proofreader checks text-to-equation consistency, summation and integration bounds, and matrix dimensions. The Auditor tests the framework itself: assumptions, conditioning, convergence, approximation validity. The Sober Checker then consolidates the three reports and re-derives each finding from the MathPix OCR text rather than the PDF, to prevent re-hallucination from the visual layout of equations.

If the paper is accompanied by a replication package, three further agents examine it: the Divergence Hunter, which looks for gaps between the paper's claims and the code; the Bug Hunter, which looks for bugs and logic errors; and the Data Archaeologist, which examines data handling and provenance. Their findings pass through a compilation stage and a verification pass that tests each claim against the actual code before anything is forwarded downstream.

The Red Team's output contains both genuine findings and the sycophancy-driven fabrications that aggression introduces, and the rest of the pipeline exists to distinguish them. Visual evidence from the attackers is rejected because they tend, under adversarial prompting, to see what fits their critique. Citations to external texts are rejected because attackers claim knowledge of sources they have not seen. Against the consolidated Red Team output, a Blue Team stage is asked to argue the paper's defense, presenting the strongest possible counterargument for each finding. An Assessment stage then rules on each finding by category, independently re-deriving anything mathematical before it passes. A final accuracy-and-fairness check marks suspected false positives for deletion. Three interposed compilation stages maintain the record without modifying its content, which keeps the decisions at each stage auditable.

The remaining stages verify the findings, write them up as a review, and verify the review in turn. The Fact Checker audits each surviving issue against the paper, confirming page numbers, quotations, and equations; the External Check audits citations of sources outside the paper, flagging details the attackers invented. The surviving list is then passed to the Reviewer, which writes the credibility assessment:

a short essay answering the question "Is it credible?", a bottom-line summary, and a brief note on future research. The assembled review passes through review-checking and citation-verification stages that treat every citation as wrong until confirmed, a legal pass for defamatory or legally risky statements, and a formatting pass. The full chain runs to more than 30 calls.

A final stage offers editorial advice. The Alchemist receives the finished review and is given two purposes, described in its own prompt as a standard conflict in academia: to help the author publish the paper and to help the author discover the truth. It first formulates a defense of the argument with as few compromises as possible, then tests whether the defense is realistic against the review's findings. If it is not, the Alchemist recommends harder choices, such as removing sections, adding caveats, and making less maximalist arguments. The Alchemist also advises the Copy-Editor on specific issues to address. The author then receives suggestions for what could be changed. The Proofreader checks for language errors.

The system is engineered for specific Gemini language models. In testing, it was found that higher capability is not always better. Many calls in the adversarial stage therefore use Gemini 2.5 Pro rather than the stronger 3.1 Pro, because the 2.5 model commits more fully to the hostile role, whereas the stronger model hedges its findings and reverts to verification.[2] Flash 2.5 handles compilation and other bureaucratic work. The verification and assessment stages, where stronger reasoning serves the task, use Gemini 3.1 Pro. [3]

Comparisons

The system can be evaluated against five existing services that produce automated peer reviews of academic papers. refine.ink, the oldest of the five, was founded by

---

[2] Ganguli et al. (2022) report that RLHF-trained models become harder to red-team as they scale, consistent with more capable models being harder to prompt out of their training defaults.

[3] The pipeline will be adjusted when Google retires the 2.5 series in June 2026. Flash 3.0 is at present not a viable replacement for Flash 2.5 on the compilation stages: it is prone to reasoning loops in which it burns tokens without producing output.

Benjamin Golub, a professor of economics at Northwestern, and Yann Calvó López, a computational-science graduate student at Harvard; it is commonly seen as the gold standard and is expensive, costing up to $50 per review. reviewer3.com, founded by Natalie Khalil, is another commercial alternative. paperreview.ai, the Stanford Agentic Reviewer by Yixing Jiang and Andrew Ng, is a free system, as is openaireview.org, from the Chicago Human+AI Lab under Chenhao Tan. The most recent alternative is coarse.ink, by David Van Dijcke, an economics PhD candidate at the University of Michigan; it appears to have effectively been reverse-engineered from refine.ink and is also open source.

The evaluation uses four papers selected by refine and criteria used by coarse in its own testing. They span neuroscience, coding theory, population genetics, and economics. None was chosen by the author of the present paper, and refine's reasons for selecting them are not documented. coarse's criteria use a language model to judge two reviews side by side on a one-to-six scale; 5.0 means the reviews are of equal quality. To control for ordering effects, each pair is scored twice with the reviews swapped, and the two scores are averaged. To reduce the influence of raw model capability on the comparison, coarse and openaireview's reviews were rerun on the higher-capability Gemini 3.1 Pro, the same model that powers Reviewer 2's main stages; the reviews from refine, openaireview, and reviewer3 are all those provided in coarse's testing materials. As shown in Table 1, the tally is 15 wins for Reviewer 2, 4 ties, and 1 loss.[4]

[4] On the population genetics paper, where Reviewer 2 scored 4.25 against refine, refine caught four mathematical errors that Reviewer 2's math stages had missed; Reviewer 2's output also contained citation errors (correct quotes attributed to wrong pages) and criticized limitations that the paper itself had already acknowledged. The pipeline has since been revised: citation verification was improved, acknowledged limitations are now handled separately, and the downstream verifiers no longer reintroduce math claims that the fact-checking stage has rejected. A further concern is the choice of judge. Using Gemini 3.1 Pro as both pipeline model and judge invites suspicion of same-model bias, but for the cells involving Reviewer 2 against coarse and openaireview, both of which also run on Gemini 3.1 Pro, that bias is present on both sides of each pairing and cancels; the 5.94 and 5.22 averages therefore reflect architectural rather than model-provenance differences. For reviewer3, paperreview.ai and refine, the underlying models are not disclosed and the argument cannot be tested directly. Gemini 3.1 Pro is nonetheless the appropriate judge given the mathematical verification these papers require. Running the same evaluation with Opus 4.6 as

Table 1
Reviewer 2 vs Alternatives

Alternative	Chaotic Balanced State	Coset Codes	Population Genetics	Targeting Interventions	Average
coarse	5.25	5.00	5.38	5.25	5.22
openaireview	6.00	6.00	5.75	6.00	5.94
paperreview	6.00	6.00	5.88	6.00	5.97
refine	5.88	5.25	4.25	5.88	5.31
reviewer3	6.00	5.88	6.00	6.00	5.97

Notes: Each cell is the average of two pairwise comparisons by a Gemini 3.1 Pro judge of Reviewer 2's review against the alternative's, with the order of the two reviews swapped between runs to control for ordering effects. Reviews are scored on a 1.0–6.0 scale with 5.0 marking parity. A cell counts as a win for Reviewer 2 when the score is above 5.25, as a tie when the score falls between 4.75 and 5.25 inclusive, and as a loss when the score is below 4.75; the quarter-point tolerance around parity reflects noise in the averaged judgment.

The major caveat is that possibly all models have been trained on these papers to some degree. Most obviously in the case of coarse, it was both trained on these papers and assessed using these criteria. Nothing is known about refine's training or what role these papers played in it or why they were chosen as examples of its capabilities. In the case of Reviewer 2, the four papers were used in the training of its math add-on, although not in the rest of the system.[5]

Availability

The system is released under the Apache 2.0 license at https://github.com/isitcredible/reviewer2. The release includes the full pipeline and every prompt. Anyone with a Gemini API key can run it on their own papers. The license also permits forks, modifications and redistribution.

---

the judge, via Claude Code, produced weaker scores (8 wins, 2 losses, 10 ties). The reviews were identical in both runs, so the difference traces to the judge: Opus flagged several of Reviewer 2's math findings as false positives, but numerical re-checks in Python on the MathPix text of each paper confirmed zero false positives across the 39 findings. Opus's failure mode is to check each equation for consistency against other equations in the same paper, which misses errors that look consistent within a derivation but contradict claims elsewhere. LLM judges exhibit systematic biases and disagree with one another (Zheng et al. 2023; Wang et al. 2023; Panickssery et al. 2024).

[5] All reviews and judge scores used in the comparison are available at https://isitcredible.com/papers/c1c8fbc4.

The system can also be run as a commercial service at isitcredible.com and through its API. The hosted service runs the current version of the pipeline, which will continue to be updated after the version released under Apache 2.0. Reports produced by the hosted service are verifiable: a report claimed to come from Reviewer 2 can be checked against the website or API, which confirms whether the document is a genuine output of the pipeline and has not been modified. This service could be useful for editors if they do not wish to set up their own internal versions of Reviewer 2.

Appendix

The full Reviewer 2 pipeline, in execution order. Each entry gives the stage code, its descriptive name, and in parentheses the Gemini model assigned; an asterisk marks an optional stage, active only for papers carrying the relevant content (mathematics or replication code) or when the user has requested the corresponding add-on (author-facing advice or copyediting).

00a Metadata (Flash 2.5)
Extracts title, authors, discipline, and research question from the PDF.

00b Metadata Clean (Flash 2.5)
Structures the extracted metadata as JSON.

00c Contributions (Flash 2.5)
Identifies the paper's claimed contributions, which are fed to the adversarial stages as targets.

01a The Breaker (Pro 2.5)
Attacks the paper's intellectual foundations. Examines whether the theoretical framework predetermines the findings, whether key premises are treated as obvious when they are actually disputed, and whether a scholar from a competing tradition would reject the entire framing. Checks that causal design labels (difference-in-differences, instrumental variables, and so on) are earned rather than merely claimed.

01b The Butcher (Pro 2.5)

Dissects the empirical machinery. Asks whether the method can actually answer the question posed, whether measures capture the theoretical constructs or just something measurable, and whether adjustment procedures (weights, matching, instruments) could introduce the bias they are supposed to remove. Checks whether robustness checks test anything threatening or merely survive trivial perturbations. Empirical papers only.

01c The Shredder (Pro 2.5)

A forensic procedural auditor. Verifies that what was claimed to have been done was actually done, based solely on the documentation provided. Checks that sample sizes match across methods, results, and tables; that timeline dates are consistent; that statistical test results yield the reported p-values; and that pre-registration claims match reported outcomes. Empirical papers only.

01d The Collector (Flash 2.5)

A forensic detail specialist. Returns to every location flagged by The Butcher and The Shredder, reads everything within two pages, and collects overlooked details: footnotes, table notes, cross-references, patterns. Does not identify new issues.

01a_2 Breaker Revisit (Pro 3.1)

For non-empirical papers only: a deeper second pass replacing The Butcher and The Shredder, with higher temperature and extended thinking to probe theoretical and mathematical arguments more aggressively.

01e Math Error Finder* (Flash 2.5)

Checks the integrity of any significant calculation that is load-bearing for the paper's contributions.

01e2 Equation Extraction* (MathPix + Flash 2.5)

Runs MathPix OCR on the PDF to extract LaTeX equations, then structures them.

01fa The Re-Deriver* (Pro 3.1)

Re-derives key results from scratch, comparing term by term against the paper's expressions. Checks sign errors, derivation leaps, change-of-variables Jacobians, and construction matching claims. When the paper says "this corresponds to construction Y," it builds Y independently and compares.

01fb The Proofreader* (Pro 3.1)

Meticulous checking of text-to-equation consistency, equation-to-equation consistency, recursion boundaries, summation and integration bounds, normalization and probability mass, and matrix or vector dimensions. Checks every footnote making mathematical claims.

01fc The Auditor* (Pro 3.1)

Audits the mathematical framework itself: assumptions, conditioning, convergence conditions, approximation validity. Tests whether the paper's mathematical apparatus rests on sound foundations. Reads The Proofreader's findings to avoid duplication, then focuses on what was missed, especially in supplementary materials.

01fd The Sober Checker* (Pro 3.1)

Consolidates findings from The Re-Deriver, The Proofreader, and The Auditor. Re-derives each finding independently and retains only those that survive independent verification. Reads only the MathPix OCR text, not the PDF, to prevent re-hallucination from visual layout.

01g The Void (Pro 2.5)

Analyzes what the paper does not say: unmeasured confounds, reverse causation, missing robustness checks, selection bias, measurement validity gaps, untested alternative explanations.

01i Divergence Hunter* (Pro 3.1)

Finds gaps between the paper's claims and its code.

01j Bug Hunter* (Pro 3.1)

Looks for bugs and logic errors in the code.

01k Data Archaeologist* (Pro 3.1)

Examines data handling, provenance, and preprocessing choices.

01l Code Compiler* (Flash 2.5)

Consolidates code findings from the three code agents.

01m Code Checker* (Pro 2.5)

Verifies consolidated code findings against the actual code.

01n Code List* (Pro 3.1)

Produces the final verified code issues list.

01o The Summarizer (Pro 3.1)

Combines all Red Team findings (adversarial, math, code) into a single consolidated list, removing duplicates.

02a Numbers (Pro 3.1)

Verifies the Red Team's mathematical claims against the paper. A bottleneck stage that uses the strongest model to prevent weaker models from overriding correct findings.

02b Compiler 1 (Flash 2.5)

First compilation pass. Strictly pass-through: reproduces findings unchanged.

02c Blue Team (Pro 3.1)

Argues the paper's defense. For each Red Team finding, presents the strongest possible counterargument. Classifies issues by type (A through G).

02d Compiler 2 (Flash 2.5)

Second compilation pass, merging Red Team and Blue Team. Strictly pass-through.

02e Assessment (Pro 3.1)

Rules on each finding: Red Team correct, Blue Team correct, or debatable. Verifies Type A errors through independent derivation.

02f Compiler 3 (Flash 2.5)

Third compilation pass, incorporating the assessment. Orders by severity. Strictly pass-through.

02g List v1 (Pro 3.1)

Final accuracy and fairness check on the consolidated list. Verifies quotes in context, suggests deletions for false positives.

03a Fact Checker (Pro 3.1)

Verifies each remaining issue against the paper, checking page numbers, quotations, and claimed equations.

03b External Check (Flash 2.5)

Audits citations of external sources. Flags invented details not stated in the PDF.

03c List v2 (Pro 3.1)

Applies corrections from the Fact Checker and External Check. Protects code-verified findings from deletion.

04a The Reviewer (Pro 3.1)

Writes the credibility assessment: the "Is It Credible?" essay (up to 700 words), "The Bottom Line" (three to five sentences), and "Future Research" (constructive proposals).

Assembly (code)

Splices the verified issues list into the review document.

05a Review Checker (Pro 3.1)

Checks the assembled review for accuracy and fairness against the PDF.

05b Citation Verifier (Pro 3.1)

Verifies every citation in the review against the PDF. Assumes all citations are wrong until confirmed.

05c Reviser (Pro 3.1)

Applies corrections from 05a and 05b. Preserves code-verified findings.

06 Legal (Flash 2.5)

Checks for defamatory or legally risky statements. Four red lines: imputed intent, fraud accusations, competence attacks, unsubstantiated absolutes.

07 Format (Pro 3.1)

Applies legal corrections, proofreads, ensures structure and LaTeX/markdown compatibility.

07b Data Editor* (Pro 3.1)

Writes a standalone code review report reflecting on how coding issues affect credibility.

08a Alchemist* (Pro 3.1)

Advises the author how to revise: identifies weak critiques, formulates a defense plan, suggests improvements.

08b Polisher* (Pro 3.1)

Rewrites the Alchemist's output in a polished editorial voice.

09a Proofreader* (Pro 3.1)

Proofreads the paper itself for spelling, grammar, and punctuation, distinguishing genuine errors from OCR artifacts.

09b Proofread Clean* (Pro 3.1)

Removes false positives from the proofreading output.

09c Copyedit* (Pro 3.1)

Produces specific revision suggestions for clarity, cohesion, consistency, and coherence.

References

Bai, Y., et al. 2022. "Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback." *arXiv preprint arXiv:2204.05862*.

Denison, C., et al. 2024. "Sycophancy to Subterfuge: Investigating Reward-Tampering in Language Models." *arXiv preprint arXiv:2406.10162*.

Dycke, N., and I. Gurevych. 2025. "Automatic Reviewers Fail to Detect Faulty Reasoning in Research Papers." *arXiv preprint arXiv:2508.21422*.

Dziri, Nouha, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. "Faith and Fate: Limits of Transformers on Compositionality." In *Advances in Neural Information Processing Systems*.

Ganguli, D., et al. 2022. "Red Teaming Language Models to Reduce Harms." *arXiv preprint arXiv:2209.07858*.

Liang, W., et al. 2024. "Monitoring AI-Modified Content at Scale." *arXiv preprint arXiv:2403.07183*.

Lin, Z. 2025. "Hidden Prompts in Manuscripts Exploit AI-Assisted Peer Review." *arXiv preprint arXiv:2507.06185*.

McCoy, R. Thomas, Shunyu Yao, Dan Friedman, Matthew D. Hardy, and Thomas L. Griffiths. 2024. "Embers of Autoregression Show How Large Language Models Are Shaped by the Problem They Are Trained to Solve." *Proceedings of the National Academy of Sciences* 121 (41): e2322420121.

Mirzadeh, Iman, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2025. "GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models." In *International Conference on Learning Representations*.

Nezhurina, Marianna, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. "Alice in Wonderland: Simple Tasks Showing Complete Reasoning Breakdown in State-of-the-Art Large Language Models." *arXiv preprint arXiv:2406.02061*.

Ouyang, L., et al. 2022. "Training Language Models to Follow Instructions with Human Feedback." *arXiv preprint arXiv:2203.02155*.

Panickssery, A., et al. 2024. "LLM Evaluators Recognize and Favor Their Own Generations." *arXiv preprint arXiv:2404.13076*.

Salewski, L., et al. 2023. "In-Context Impersonation Reveals Large Language Models' Strengths and Biases." *arXiv preprint arXiv:2305.14930*.

Shanahan, M., K. McDonell, and L. Reynolds. 2023. "Role-Play with Large Language Models." *arXiv preprint arXiv:2305.16367*.

Sharma, M., et al. 2023. "Towards Understanding Sycophancy in Language Models." *arXiv preprint arXiv:2310.13548*.

Sharma, M., M. McCain, A. Douglas, and D. Duvenaud. 2026. "Who's in Charge? Disempowerment Patterns in Real-World LLM Usage." *arXiv preprint arXiv:2601.19062*.

Wang, P., et al. 2023. "Large Language Models Are Not Fair Evaluators." *arXiv preprint arXiv:2305.17926*.

Wei, A., N. Haghtalab, and J. Steinhardt. 2023. "Jailbroken: How Does LLM Safety Training Fail?" *arXiv preprint arXiv:2307.02483*.

Zheng, L., et al. 2023. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." *arXiv preprint arXiv:2306.05685*.