PaperAudit-Bench: Benchmarking Error Detection in Research Papers for Critical Automated Peer Review

⚠ **WARNING: This paper contains intentionally AI-corrupted content for research use only.**

Anonymous ACL submission

Abstract

Large language models can generate fluent peer reviews, yet their assessments often lack sufficient critical rigor when substantive issues are subtle and distributed across a paper. In this paper, we introduce **PaperAudit-Bench**, which consists of two components: **(1) PaperAudit-Dataset**, an error dataset covering both errors identifiable within individual sections and those requiring cross-section reasoning, designed for controlled evaluation under long-context settings; and **(2) PaperAudit-Review**, an automated review framework that integrates structured error detection with evidence-aware review generation to support critical assessment. Experiments on PaperAudit-Bench reveal large variability in error detectability across models and detection depths, highlighting the difficulty of identifying such errors under long-context settings. Relative to representative automated reviewing baselines, incorporating explicit error detection into the review workflow produces systematically stricter and more discriminative evaluations, demonstrating its suitability for peer review. Finally, we show that the dataset supports training lightweight LLM detectors via SFT and RL, enabling effective error detection at reduced computational cost.

1 Introduction

With the increasing use of large language models (LLMs) in scientific writing and research assistance, automated peer review has become a growing topic in AI for Science. Recent advances in supervised fine-tuning (SFT) and reinforcement learning (RL) have improved LLM reasoning and alignment, enabling coherent and well-structured long-form analysis (Fu et al., 2025; Tu et al., 2025a,b), and prior work shows that such models can generate fluent review texts whose surface form increasingly resembles that of human reviewers (Zhou et al., 2024; Zhu et al., 2025b).

Despite these observations, the scope and characteristics of such limitations remain difficult to assess, largely due to the absence of systematic benchmarks for *paper-level* errors. Such errors may appear locally within individual sections or manifest as inconsistencies that only become evident when comparing distant parts of a paper, while large-scale collections of real papers with author-acknowledged mistakes remain scarce. In practice, current AI reviewers often lack mechanisms for systematically probing such issues, which may result in biased or overly permissive review outcomes (Wu et al., 2024; Dycke and Gurevych, 2025; Li et al., 2025a; Zhu et al., 2025a).

To address this gap, we introduce **PaperAudit-Bench**, a comprehensive benchmark for auditing scientific papers and systematically evaluating the critical assessment behavior of automated peer review systems. PaperAudit-Bench consists of two tightly coupled components: **I. PaperAudit-Dataset**, which constructs a ***paper-level error corpus*** by injecting diverse, sparse, and realistically distributed errors into high-quality conference papers through model-driven synthetic editing, covering major risk dimensions of scientific manuscripts; **II. PaperAudit-Review**, a unified automated reviewing framework consisting of two workflows: a structured ***error detection workflow*** operating under three modes (fast single-pass, standard with document-level *global memory*, and deeper multi-step), and an ***evidence-aware review workflow*** for analyzing the error sensitivity and criticality of AI reviewers.

Extensive experiments demonstrate that (1) state-of-the-art LLMs exhibit substantial variability in detecting errors, with uneven sensitivity across error types and paper sections; (2) structured detection and document-level global memory are essential for exposing distributed errors and improving

Code is available at: https://anonymous.4open.science/r/PaperAudit-43C2.

Benchmark / Dataset	Source	Data Scale	Error Types	Detect Mode	Review Integrate	Train Detectors
Aspect-Perturb (Li et al., 2025a)	Synthetic	508 papers (79k edits)	S	-	✓	✗
SPOT (Son et al., 2025)	Real	83 papers (91 errs)	M	S	✗	✗
FLAWS (Xi et al., 2025)	Synthetic	713 papers (1 err / paper)	S	S	✗	✗
PRISMM-Bench (Selch et al., 2025)	Real	242 papers (262 errs)	S	S	✗	✗
DeepReview (Zhu et al., 2025b)	Real	13k papers	-	-	✓	✗
PaperAudit-Bench (Ours)	Synthetic	220 papers (>10 errs / paper)	M	M	✓	✓

Table 1: Comparison of PaperAudit-Bench with existing benchmarks and review datasets. **M** denotes multi-type or multi-stage support, **S** denotes single-type or single-stage support, and **-** denotes not applicable.

error coverage; (3) the proposed dataset enables SFT and RL training of lightweight detectors, yielding compact models with competitive detection performance and improved output discipline; and (4) integrating explicit error detection into the review process systematically shifts AI reviewers toward more critical and discriminative evaluations grounded in identifiable technical and argumentative issues, particularly with respect to technical quality and argumentative rigor.

Overall, our goal is not to replace human reviewers, but to foster more critical and evidence-grounded automated peer review through controlled benchmarks and structured error detection.

2 Related Works

AI-assisted Scientific Research. Recent work explores AI-assisted scientific research across the research lifecycle, including idea generation, experimentation, and paper writing (Ren et al., 2025; Chen et al., 2025b). While end-to-end research systems and scientific agents continue to evolve (Lu et al., 2024; Yamada et al., 2025; Weng et al., 2025b), autonomously conducting complex scientific reasoning and verification remains challenging (Chen et al., 2025a; Kon et al., 2025). In contrast, LLMs have shown strong ability in drafting academic papers when provided with existing materials, raising concerns about the reliability of AI research (Ren and Wang, 2025; Hou et al., 2025).

AI-based Paper Review and Error Detection. Recent studies on automated peer review primarily focus on generating fluent and structured reviews using LLMs, often trained via SFT or RL (Zhu et al., 2025b; Zeng et al., 2025). However, existing evidence indicates that such reviewers frequently miss substantive errors in papers and are sensitive to perturbations in rebuttals, leading to unreliable judgments (Wu et al., 2024; Li et al., 2025c,a; Dycke and Gurevych, 2025). To address this issue, several benchmarks evaluate error detection using real but scarce error cases (Son et al., 2025) or synthetic settings with localized verification objectives (Xi et al., 2025; Selch et al., 2025). As summarized in Table 1, existing resources either lack controlled, distributed errors or do not integrate error detection into the reviewing process. PaperAudit-Bench complements these efforts by enabling document-level evaluation of multi-type errors, together with explicit detection workflows and error-aware review. A more detailed discussion of related benchmarks and review systems is provided in the Appendix A.

3 PaperAudit-Dataset: Benchmarking Error Detection for Research Papers

In this section, we introduce **PaperAudit-Dataset**, a dataset for evaluating error detection in AI research paper. We describe its construction pipeline, error injection process, and key characteristics.

3.1 Paper Collection and Preprocessing

PaperAudit-Dataset is constructed from research papers accepted at the three top AI conferences: ***ICLR***, ***ICML***, and ***NeurIPS*** in ***2025***. We focus on ***oral*** papers, which are generally regarded as high-quality and contain relatively few obvious errors, ensuring that injected corruptions constitute the primary source of errors in the dataset. Papers are collected from ***OpenReview*** and filtered using a lightweight length-based criterion to exclude extremely long documents, which may introduce instability in long-context processing. After filtering, the dataset comprises 107 ICLR, 67 ICML, and 46 NeurIPS papers in PDF format.

Each paper is converted into a *structured, section-aware JSON representation* designed for multimodal language models. The representation consists of an ordered sequence of text and figure blocks, each annotated with a semantic section label, providing a standardized input format for downstream error injection, detection, and review. Implementation-level details of PDF parsing, multimodal extraction, section labeling, and rule-based cleanup are provided in Appendix C.1.

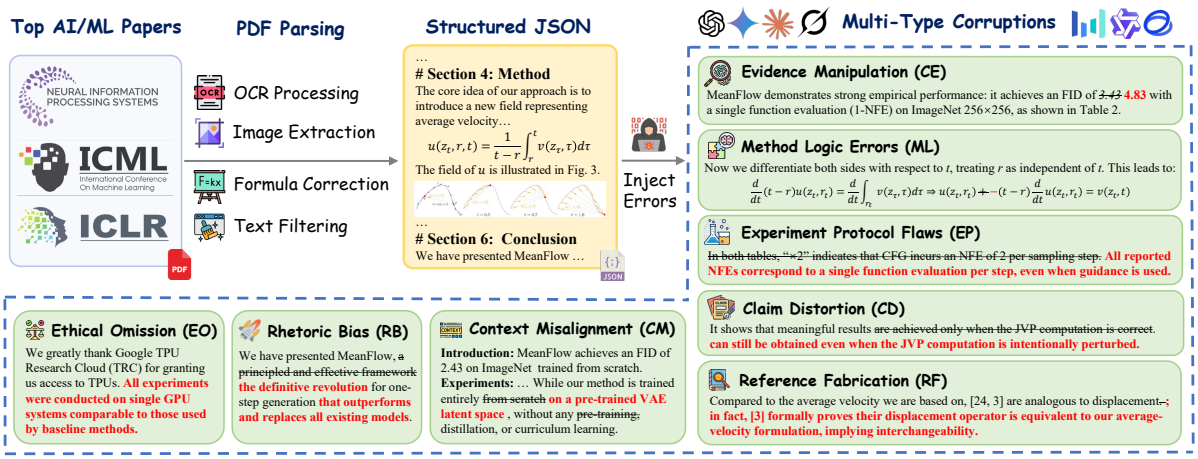
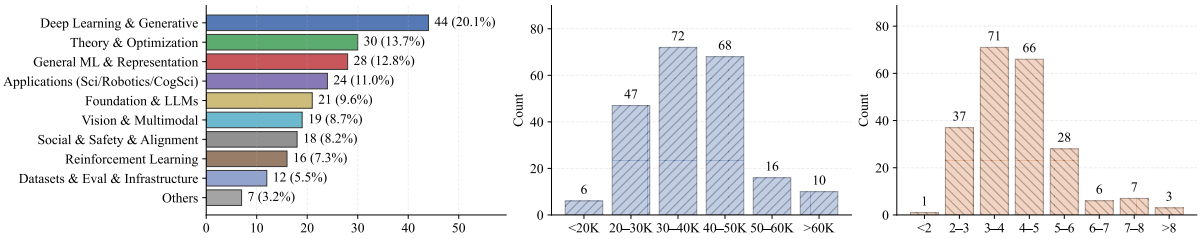


Figure 1: Overview of ***PaperAudit-Dataset*** and examples of errors injected through *model-driven synthetic editing*.



(a) Research Area Distribution (b) Input Token Length per Paper (c) Error Density per 10K Tokens

Figure 2: Statistics of ***PaperAudit-Dataset***. (**a**) Distribution of research areas covered in PaperAudit-Dataset. (**b**) Distribution of input token lengths per paper, estimated under the GPT-5 default configuration. (**c**) Distribution of injected error density, measured as errors per 10K tokens, computed from GPT-5-generated corrupted papers.

3.2 Synthetic Error Injection

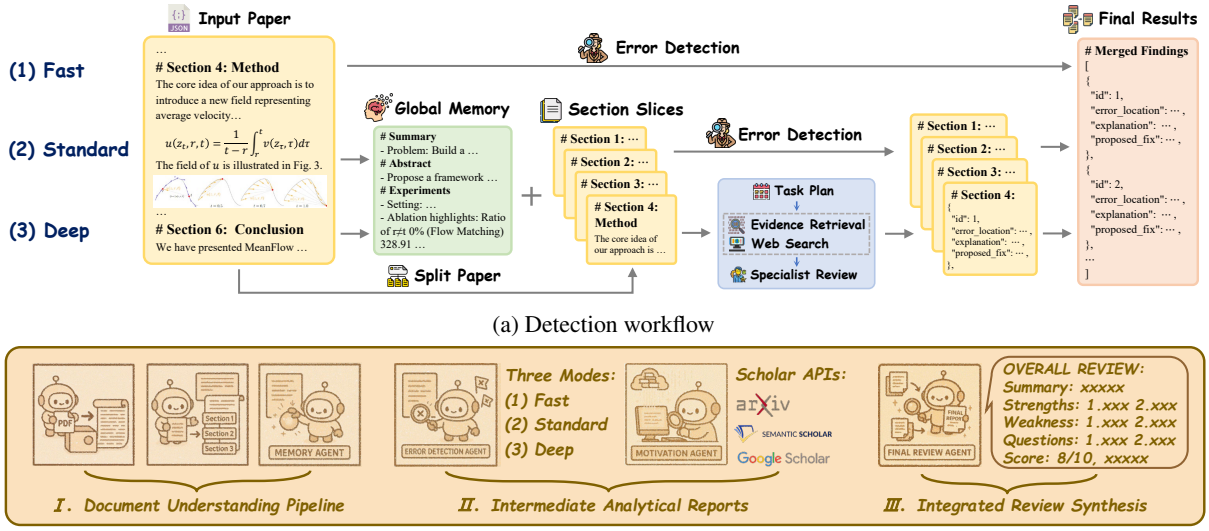
To enable systematic evaluation of paper auditing capabilities, we adopt a *model-driven synthetic editing* strategy to inject realistic and controllable paper-level errors. For each source paper, multiple LLMs are used to generate different corrupted versions, with each model prompted to inject 10–20 errors distributed across multiple locations and sections, covering diverse error types to simulate realistic auditing challenges.

Inspired by established guidelines on research integrity (National Academies of Sciences et al., 2017; Zhang and Abernethy, 2025), we identify and design eight categories of errors commonly observed in papers, spanning from unintentional mistakes to more subtle forms of misrepresentation and misconduct. Detailed descriptions of these error categories are provided in Table 9, and failed or implausible synthesized errors are filtered via additional quality control in Appendix C.2. Importantly, many errors require paper-level reasoning across **distant sections** of a paper, rather than inspection of isolated text segments, such as verifying claims in the abstract against evidence in the experiments.

3.3 Dataset Characteristics

PaperAudit-Dataset is designed as a challenging and realistic benchmark for paper-level auditing. Figure 1 summarizes the dataset construction pipeline from paper collection and preprocessing to error injection, with representative examples for each error category. The dataset comprises **220 papers**, with **approximately 15 injected errors** per paper on average, of which **over 10 valid errors** are retained after filtering and quality control.

Beyond the construction process, the dataset exhibits several intrinsic properties that characterize the difficulty of paper-level auditing. As shown in Figure 2, the dataset covers a broad range of AI research subfields. Moreover, paper-level error detection inherently constitutes a **long-context analysis problem**, as individual papers span tens of thousands of tokens and require reasoning over information distributed across distant sections of the document. At the same time, injected errors are intentionally **sparse**, occurring only a few times per 10K tokens on average, closely reflecting real-world reviewing scenarios in which critical issues are rare and easily overlooked.



(a) Detection workflow

(b) Review workflow

Figure 3: Overview of the **PaperAudit-Review Framework**. The detection workflow supports three levels of analytical depth, while the review workflow builds upon the detection results to synthesize critical assessments.

4 PaperAudit-Review: Error Detection and Critical Review Framework

In this section, we introduce **PaperAudit-Review**, a unified framework that couples structured error detection with critical peer review. It comprises an error detection component with varying analytical depth and a review component that integrates detected errors to support evidence-aware review.

4.1 Detection Workflow

We propose three error detection modes with increasing analytical depth (Figure 3a), ranging from single-pass analysis to section-wise and multi-agent detection, providing authors with error findings of varying granularity to highlight potential risks and guide revision.

Fast Mode: Single-Pass Global Detection. The ***Fast*** mode performs a single-pass scan over the entire paper, treating the document as a whole without explicit section decomposition. It prioritizes prominent and high-level issues, particularly cross-section support mismatches, and is designed for efficient large-scale screening with minimal computational overhead.

Standard Mode: Section-Aware Detection with Global Context. The ***Standard*** mode introduces explicit section awareness and a shared global memory. The paper is decomposed into semantic sections, each reviewed independently while being conditioned on a global memory of the full document. This design enables finer-grained detection within individual sections while supporting cross-section reasoning through shared global memory, making it effective for identifying inconsistencies that are distributed across distant parts of the paper.

Deep Mode: Plan-Driven Multi-Agent Detection. The ***Deep*** mode further extends the Standard setting by incorporating explicit planning and specialized analysis. It decomposes the review process into targeted tasks spanning multiple sections, optionally retrieves additional supporting evidence, and assigns each task to domain-specific specialist agents. This mode improves coverage and systematic exploration of diverse error modes, more closely approximating human reviewing behavior, at the cost of higher computational complexity.

Formal definitions, algorithmic details, and prompt specifications for all three detection modes are provided in Appendix C.3.

4.2 Training Lightweight Detectors

We train a lightweight detector via a two-stage pipeline combining SFT and RL, both to replace the large-scale LLM detector in Section 4.1, the main computational bottleneck, and to examine whether training on synthesized errors from PaperAudit-Dataset improves error detection performance.

SFT with Structured Data. We first perform SFT on structured data derived from PaperAudit-Dataset. The training set is defined as $D_{\text{train}} = \{(x_{\text{synth_section}}, y)\}$, where $x_{\text{synth_section}}$ denotes a *corrupted paper section* and y consists of *struc-*

tured annotations specifying the error type, location, and explanation. This stage trains the model to map corrupted content to structured error descriptions, equipping it with foundational error reasoning capability and reliable output format control.

RL with Multi-Dimensional Rewards. To further improve detection accuracy and reduce redundant outputs, we apply RL using *Group Relative Policy Optimization (GRPO)* (Shao et al., 2024). We design a multi-dimensional reward function that jointly captures detection accuracy, output conciseness, and abnormal behavior penalties:

$$r(x, a, y) = w_p \cdot \text{Precision}(a, y) + w_c \cdot \text{Conciseness}(a) + \sum_{k=1}^{K} \mathcal{P}_k(a, y), \quad (1)$$

where a denotes the model detection answer, $\text{Precision}(a, y)$ and $\text{Conciseness}(a)$ measure semantic alignment with expert annotations and penalize verbose or redundant responses, respectively, both taking values in $[0, 1]$ and evaluated using an external judge model, with w_p and w_c denoting the weighting coefficients; the penalty term $\mathcal{P}_k(a, y)$ further penalizes abnormal outputs, including empty responses (-0.8) and violations of the required JSON format (-0.5). After RL, reasoning accuracy and output discipline improve, yielding compact detectors for the workflow.

4.3 Applying Error Detection to Peer Review

Beyond providing actionable feedback to authors, error detection also enhances the **peer review** process by supporting more critical analysis. Figure 3b illustrates the overall review workflow, which integrates the proposed detection framework to enable **critical** and structured reviewing. The process begins with document understanding, where the input paper is parsed into a structured representation and summarized into a global memory. Based on this representation, a **baseline review** is first generated to provide an initial assessment. In parallel, two analytical processes are conducted: ***fine-grained error detection*** using the proposed detection workflow, and ***novelty analysis*** that queries scholarly APIs to identify related work and assess similarity and originality. The resulting error and motivation reports are then used to **refine and adjust** the baseline review, resulting in an integrated final review that combines critical findings, strengths and weaknesses, and an overall assessment.

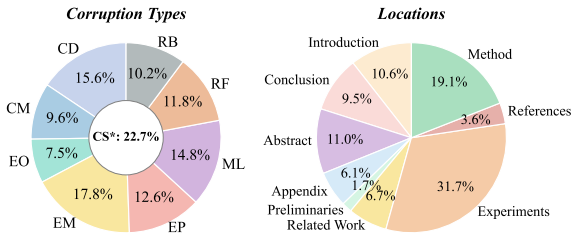


Figure 4: Distribution of injected error types and their locations in the *NeurIPS* subset of *PaperAudit-Dataset*. The statistics are averaged over corrupted papers generated by eight synthesis models.

5 Evaluation

We evaluate *PaperAudit* along three progressively connected research questions:

- **RQ1:** How detectable are paper-level errors, and how does detectability vary across models, error types, and document locations?
- **RQ2:** Does explicit error detection lead to more critical AI-based peer reviews?
- **RQ3:** Can the proposed benchmark support training lightweight detectors?

5.1 RQ1: Error Detectability across Models

We first evaluate the intrinsic detectability of paper-level errors to assess whether *PaperAudit-Dataset* is non-trivial and discriminative.

Dataset and Error Distribution. For computational feasibility, we evaluate on the *NeurIPS* subset of *PaperAudit-Dataset*, which contains 46 original papers with synthetic corruptions generated by eight synthesis models. Figure 4 summarizes error types and locations. Errors span diverse categories and concentrate in technical sections (*Method*, *Experiments*), while also appearing in high-level sections such as the *Abstract* and *Conclusion*. We additionally annotate *cross-section consistency* errors (CS*), which require global context and cannot be resolved by isolated section-level inspection.

Evaluation Metrics. We evaluate error detection using three complementary paper-level metrics capturing coverage, precision, and their trade-off. Error Coverage (EC) measures coverage of injected ground-truth (GT) errors, while Finding Precision (FP) measures how many reported findings correspond to injected errors, defined as:

$$\text{EC} = \frac{\#\text{Matched Errors}}{\#\text{GT Errors}}, \quad \text{FP} = \frac{\#\text{Matched Findings}}{\#\text{All Findings}},$$

Here, matched errors and findings are identified

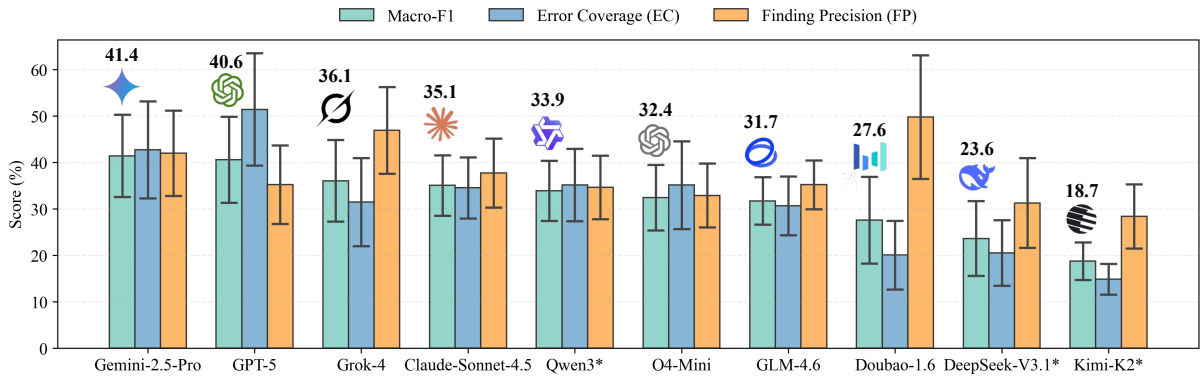


Figure 5: Detection performance of different models under the *Fast* mode, evaluated using Macro-F1, Error Coverage (EC), and Finding Precision (FP). Bars show the mean performance with standard deviation across data synthesized by eight synthesis models. Models marked with * denote non-multimodal models where visual content is removed from the input papers. We use *Qwen3-235B-A22B* instead of *Qwen3-VL-235B-A22B*, as the latter does not reliably produce outputs in the required parsable format. Unless otherwise specified, all models are evaluated without explicit reasoning mechanisms, except for *o4-mini*.

via ***LLM-as-a-judge*** with GPT-5.1, which adjudicates semantic equivalence between detected findings and GT errors. For each paper, we compute an F1 score as the harmonic mean of EC and FP, and report Macro-F1 by averaging across papers:

$$\text{Macro-F1} = \frac{1}{N}\sum_{i=1}^{N}\frac{2\cdot \text{EC}_i \cdot \text{FP}_i}{\text{EC}_i + \text{FP}_i}.$$

Unlike standard classification settings, unmatched findings are not necessarily false positives, as even high-quality papers may surface reasonable but non-injected issues whose validity is difficult to conclusively determine. Accordingly, some analyses place greater emphasis on Error Coverage (EC), which reflects the detector's ability to identify injected errors. Potential over-coverage is primarily examined through qualitative *case study* analysis.

Detection Models Exhibit Distinct Coverage-Precision Trade-offs. Figure 5 summarizes detection performance under the *Fast* mode. Overall, Gemini-2.5-Pro and GPT-5 achieve the strongest performance, attaining the highest Macro-F1 scores (41.4% and 40.6%, respectively). GPT-5 exhibits the highest Error Coverage (51.4%), which is partly attributable to its tendency to produce a larger number of findings in a single-pass scan, while Gemini-2.5-Pro demonstrates a more balanced behavior with comparatively higher Finding Precision. In contrast, lightweight or conservative models such as Doubao-Seed-1.6 and Kimi-K2 generate substantially fewer findings under long-context inputs, resulting in markedly lower error coverage. Detailed per-model and per-synthesis statistics are reported in Appendix B.1.

Detect Model \ Synth Model	GPT-5	O4-Mini	Gemini-2.5-Pro	Claude-Sonnet-4.5	Grok-4	Avg.
Avg.	**0.62**	**0.57**	**0.43**	**0.45**	**0.43**	**0.50**
GPT-5	0.75	0.66	0.55	0.58	0.56	**0.62**
O4-Mini	0.53	0.57	0.38	0.35	0.39	**0.44**
Gemini-2.5-Pro	0.68	0.56	0.47	0.46	0.45	**0.52**
Claude-Sonnet-4.5	0.53	0.51	0.40	0.51	0.41	**0.47**
Grok-4	0.59	0.56	0.35	0.35	0.37	**0.44**

Figure 6: Error Coverage under the Standard mode across synthesis and detection model combinations. Rows correspond to detection models and columns to synthesis models. Row-wise and column-wise averages summarize overall detection capability and error detectability, respectively.

Error Detectability Varies across Synthesis Models and Reflects Detector-Specific Biases. Figure 6 reports error coverage under the *Standard* mode across different synthesis and detection model combinations, which allows detection models to analyze the full paper with section-level context. Errors synthesized by GPT-5 are the most detectable on average (0.62), while GPT-5 also achieves the strongest detection performance as a detector (0.62), whereas Grok-4 produces more concealed errors (0.43) and exhibits lower detection performance (0.44). Additionally, detection models often achieve higher precision on errors

synthesized by the same model, which may be attributed to model-specific biases or overlap between generation and detection behaviors.

Error Detectability Depends on Error Semantics and Document Location. We analyze how error detectability varies across different corruption types and paper sections on the ICML subset, covering multiple detection models, synthesis models, and detection depths. Overall, error coverage consistently improves with increased analytical depth, while exhibiting systematic variation across error categories and locations. Errors involving subtle over-claiming or rhetorical exaggeration (e.g., Claim Distortion and Rhetoric Bias) are generally harder to detect, whereas errors grounded in explicit inconsistencies, logical flaws, or missing disclosures (e.g., Context Misalignment, Method Logic, and Ethical Omission) are more readily identified. We further observe that removing *global memory* degrades detection performance, particularly for cross-section consistency errors, highlighting the importance of shared document-level context. Detection difficulty also varies by paper section, with errors in abstract being consistently the most challenging to uncover. Detailed quantitative and qualitative results and section-wise analyses are provided in Appendix B.2 and D.1.

5.2 RQ2: Error-Aware AI Peer Review

We next examine whether explicit error detection can be effectively integrated into peer review workflows, beyond standalone detection accuracy. Following the workflow in Section 4.3, we adopt a widely used AI reviewing pipeline as the baseline, and refine them using error reports and motivation reports to produce *PaperAudit-Review*, an error-aware critical reviewer. For comparison, we evaluate *DeepReview* (Zhu et al., 2025b), a representative multi-perspective reviewing framework that aggregates feedback from multiple reviewer roles.

PaperAudit-Review Produces More Critical and Discriminative Peer Reviews. We first evaluate reviewer behavior in terms of numerical scores. All reviewers are evaluated on a 10-point scale for *Novelty*, *Technical Quality*, and *Clarity*, together with an overall score in Table 2. Across all methods, synthetically corrupted papers consistently receive lower scores than their original counterparts, with the largest declines observed in

https://github.com/NeuroDong/Ai-Review

Reviewer	Synth	Novelty	Tech	Clarity	Overall
Baseline-Review					
Gemini-2.5-Pro	***Origin***	8.70	8.30	8.09	8.20
	GPT-5	8.70	7.40	6.96	7.78
	O4-Mini	8.52	7.43	6.96	7.48
GPT-5	***Origin***	7.61	6.63	7.19	7.27
	GPT-5	7.30	5.75	6.17	6.63
	O4-Mini	6.97	5.79	6.09	6.41
PaperAudit-Review					
Gemini-2.5-Pro	***Origin***	8.51	6.31	6.51	6.81
	GPT-5	8.48	5.55	5.45	6.31
	O4-Mini	8.43	5.73	5.69	6.07
GPT-5	***Origin***	7.61	6.15	6.55	6.93
	GPT-5	7.29	5.45	5.89	6.45
	O4-Mini	6.99	5.55	5.76	6.28
DeepReview (Zhu et al., 2025b)					
Gemini-2.5-Pro	***Origin***	7.90	7.68	7.75	7.53
	GPT-5	7.71	6.63	7.21	6.82
	O4-Mini	7.74	6.70	6.61	6.57
GPT-5	***Origin***	7.59	6.91	7.64	7.35
	GPT-5	7.39	6.07	6.74	6.65
	O4-Mini	7.12	6.07	6.55	6.54

Table 2: Average review scores on ICML branch of PaperAudit-Dataset.

technical quality and clarity. Compared to baseline reviews, PaperAudit-Review assigns systematically stricter scores due to explicit error evidence, an effect that also extends to original papers by surfacing minor issues. Accordingly, while the critical mode enforces rigorous scrutiny and reliably differentiates corrupted papers, its scores should not be interpreted as absolute judgments, consistent with prior observations that AI reviewing systems should prioritize the substance of review feedback over final numerical scores (Jin et al., 2024). Instead, its primary value lies in producing more demanding and informative review comments, as further analyzed qualitatively in Appendix D.2.

In addition, we analyze which error categories are most strongly associated with score reductions for GPT-5. We find that contradictions in method logic and distorted or overstated claims are the primary drivers of score drops, whereas presentation-level or protocol-related issues often do not directly affect numerical scores; detailed statistics and examples are provided in Appendix B.5.

PaperAudit-Review Exhibit Improved Alignment with Human Judgments. We evaluate alignment between AI reviewers and human judgments on 50 randomly sampled ICLR 2026 submissions using both score-based and coverage-based

Panel A: Score-based Alignment with Human Reviewers				
Method	**R-MSE** ↓	**Spearman** ↑	**Kendall** ↑	**P-Acc** ↑
Baseline	0.152	0.119	0.085	0.545
PaperAudit	**0.148**	**0.142**	**0.109**	**0.557**
DeepReview	0.160	0.075	0.057	0.529
Panel B: Coverage-based Alignment with Human Reviewers				
Method	**Str-Cov** ↑	**Weak-Cov** ↑	**AI-Extra** ↓	**Sym-Cov** ↑
Baseline	0.876	0.568	0.495	0.433
PaperAudit	**0.886**	**0.591**	**0.468**	**0.444**
DeepReview	0.844	0.482	0.488	0.412

Table 3: Alignment with human reviewers on 50 ICLR 2026 submissions using GPT-5 reviewers. **Panel A** reports score-based alignment, while **Panel B** reports coverage-based alignment of review text, evaluated using Gemini-2.5-Pro as an external judge.

criteria. Score-based alignment (Table 3, Panel A) measures consistency in relative rankings rather than absolute scores, with human scores averaged across reviewers, while coverage-based alignment (Panel B) assesses whether automated reviews attend to the same substantive strengths and weaknesses as human reviewers by measuring recall, additional major points, and symmetric coverage similarity. Across both views, the refined *PaperAudit* reviewer consistently outperforms its baseline and *DeepReview*, achieving lower rank error, higher correlation and agreement, and improved coverage of human-identified issues. Details of the alignment metrics and evaluation protocols are provided in Appendix B.3 and Appendix B.4.

Nevertheless, absolute alignment remains limited, indicating a persistent gap between LLM-based and human reviewers. Overall, error-aware refinement improves relative alignment, positioning it as tools for supporting critical feedback rather than replacing human scoring.

5.3 RQ3: Evaluating Lightweight Detectors

Finally, we examine whether *PaperAudit-Dataset* supports training compact detectors, enabling deployment beyond frontier-scale models.

Dataset and Settings. We train lightweight detectors on 5,161 synthetic instances from the ICLR and NeurIPS branches, covering eight synthesis models. Evaluation is conducted on the ICML branch, mainly synthesized by *GPT-5* and *O4-Mini*, comprising 1,209 section-level corrupted samples with an average of 1.11 errors per sample. We perform post-training on *Llama3.2-3B*, *Qwen3-8B*, and *Qwen3-14B* using SFT with LoRA, followed by RL with $w_p = 0.6$, $w_c = 0.4$, and KL coefficient $\beta = 0.3$. *Qwen3-80B* is used as the judge for reward computation and metric evaluation.

	Detection@1			**Detection@all**		
Model	EC	FP	F1	EC	FP	F1
Gemini-2.5-Pro	48.0	53.8	50.6	75.4	27.3	40.1
Claude-Sonnet-4.5	45.8	51.0	48.3	**82.0**	20.5	32.8
Qwen3-235B-A22B	43.5	48.6	45.9	79.1	20.6	32.6
DeepSeek-V3.1	29.0	32.3	30.6	67.1	18.6	29.1
Llama3.2-3B	6.4	7.1	6.7	34.2	5.6	10.1
+ SFT	27.3	30.4	28.7	36.8	41.0	38.8
+ SFT + RL	30.4	33.9	32.1	37.8	42.1	39.8
Qwen3-8B	13.9	20.5	16.6	40.5	38.5	39.5
+ SFT	38.4	42.8	40.5	43.1	47.6	45.2
+ SFT + RL	52.3	58.3	55.1	56.6	63.1	59.7
Qwen3-14B	16.9	20.5	18.5	43.7	40.5	42.0
+ SFT	42.4	47.2	44.7	48.5	52.7	50.5
+ SFT + RL	**52.9**	**59.0**	**55.8**	57.1	**63.7**	**60.2**

Table 4: Detection performance. *Detection@1* restricts the model to report only one error with the highest confidence by prompting, while *Detection@all* allows the model to report all errors without constraint.

Performance. Table 4 reports the detection performance. Across all three lightweight backbones, both SFT and subsequent RL consistently improve detection quality. Notably, Qwen3-8B after SFT+RL achieves performance comparable to, and in some cases exceeding, that of much larger general-purpose models, which exhibit high EC but low FP under *Detection@all*, indicating a tendency toward over-detection of weakly supported issues. In contrast, our post-trained detectors achieve a more balanced EC–FP trade-off, leading to higher overall F1 scores and more focused error identification. Figure 8 further shows that RL increases reward while reducing response length, reflecting more accurate and concise detection behavior.

6 Conclusion

We present PaperAudit-Bench as a unified benchmark for evaluating paper-level error detection and its interaction with automated peer review. Our experiments reveal persistent challenges in identifying sparse, distributed errors and illustrate how structured detection changes review behavior in a more conservative and evidence-grounded manner. We hope this work provides a useful reference point for future studies on evaluating and designing error-aware automated reviewing systems.

Limitations

We summarize key practical limitations related to the dataset construction, evaluation protocol, and deployment setting.

- **Synthetic errors.** Errors in PaperAudit-Dataset are synthetically injected rather than originating from real authors. While enabling controlled evaluation, such errors cannot fully capture naturally occurring issues, and unannotated flaws in original papers make precision-based evaluation inherently imperfect. Moreover, collecting comprehensive and reliable annotations of author-acknowledged real errors remains challenging and resource-intensive.
- **Potential over-criticality.** Deeper detection settings, such as the Deep mode, may identify fine-grained or borderline issues that reflect stricter scrutiny rather than clear errors, potentially leading to over-critical assessments without careful calibration. An illustrative example is provided in Table 13 in Appendix D.1.
- **Prompt and configuration sensitivity.** The behavior of error detection and review refinement depends on prompt design and overall system configuration. While we follow a fixed and transparent setup throughout all experiments, alternative design choices or calibration strategies may lead to different trade-offs between coverage and strictness.

Ethical Considerations

This work uses LLMs to synthetically modify research papers and generate automated reviews solely for benchmark construction and evaluation. We **DO NOT** encourage paper manipulation or academic misconduct, **NOR DO** we advocate replacing human reviewers in real peer-review processes. The benchmark is intended only to assess and characterize the capabilities and limitations of current LLMs in error detection and review. LLM-based agents should be used, at most, for pre-submission self-auditing and review self-improvement, with all final judgments reserved for human experts.

References

Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. 2025a. Mlr-bench: Evaluating ai agents on open-ended machine learning research. *arXiv preprint arXiv:2505.19955*.

Qiguang Chen, Mingda Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, and 1 others. 2025b. Ai4research: A survey of artificial intelligence for scientific research. *arXiv preprint arXiv:2507.01903*.

Nils Dycke and Iryna Gurevych. 2025. Automatic reviewers fail to detect faulty reasoning in research papers: A new counterfactual evaluation framework. *arXiv preprint arXiv:2508.21422*.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*.

Aniketh Garikaparthi, Manasi Patwardhan, Lovekesh Vig, and Arman Cohan. 2025. Iris: Interactive research ideation system for accelerating scientific discovery. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 592–603.

Junyi Hou, Andre Lin Huikai, Nuo Chen, Yiwei Gong, and Bingsheng He. 2025. Paperdebugger: A plugin-based multi-agent system for in-editor academic writing, review, and editing. *arXiv preprint arXiv:2512.02589*.

Xiang Hu, Hongyu Fu, Jinge Wang, Yifeng Wang, Zhikun Li, Renjun Xu, Yu Lu, Yaochu Jin, Lili Pan, and Zhenzhong Lan. 2025. Nova: An iterative planning framework for enhancing scientific innovation with large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 21330–21359.

Yiqiao Jin, Qinlin Zhao, Yiyang Wang, Hao Chen, Kaijie Zhu, Yijia Xiao, and Jindong Wang. 2024. Agentreview: Exploring peer review dynamics with llm agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, page 1208–1226.

Patrick Tser Jern Kon, Jiachen Liu, Xinyi Zhu, Qiuyi Ding, Jingjia Peng, Jiarong Xing, Yibo Huang, Yiming Qiu, Jayanth Srinivasa, Myungjin Lee, and 1 others. 2025. Exp-bench: Can ai conduct ai research experiments? *arXiv preprint arXiv:2505.24785*.

Jiatao Li, Yanheng Li, Xinyu Hu, Mingqi Gao, and Xiaojun Wan. 2025a. Where do llms go wrong? diagnosing automated peer review via aspect-guided multi-level perturbation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 1572–1581.

Long Li, Weiwen Xu, Jiayan Guo, Ruochen Zhao, Xingxuan Li, Yuqian Yuan, Boqiang Zhang, Yuming Jiang, Yifei Xin, Ronghao Dang, and 1 others. 2025b. Chain of ideas: Revolutionizing research via novel idea development with llm agents. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8971–9004.

Ruochi Li, Haoxuan Zhang, Edward Gehringer, Ting Xiao, Junhua Ding, and Haihua Chen. 2025c. Unveiling the merits and defects of llms in automatic review generation for scientific papers. *arXiv preprint arXiv:2509.19326*.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

National Academies of Sciences, Medicine, Global Affairs, and Committee on Responsible Science. 2017. *Fostering integrity in research*. National Academies Press.

Yansheng Qiu, Haoquan Zhang, Zhaopan Xu, Ming Li, Diping Song, Zheng Wang, and Kaipeng Zhang. 2025. Ai idea bench 2025: Ai research idea generation benchmark. *arXiv preprint arXiv:2504.14191*.

Jing Ren and Weiqi Wang. 2025. Assisting research proposal writing with large language models: Evaluation and refinement. *arXiv preprint arXiv:2509.09709*.

Shuo Ren, Pu Jian, Zhenjiang Ren, Chunlin Leng, Can Xie, and Jiajun Zhang. 2025. Towards scientific intelligence: A survey of llm-based scientific agents. *arXiv preprint arXiv:2503.24047*.

Kai Ruan, Xuan Wang, Jixiang Hong, Peng Wang, Yang Liu, and Hao Sun. 2024. Liveideabench: Evaluating llms' divergent thinking for scientific idea generation with minimal context. *arXiv preprint arXiv:2412.17596*.

Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

Lukas Selch, Yufang Hou, M Jehanzeb Mirza, Sivan Doveh, James Glass, Rogerio Feris, and Wei Lin. 2025. Prismm-bench: A benchmark of peer-review grounded multimodal inconsistencies. *arXiv preprint arXiv:2510.16505*.

Chenyang Shao, Dehao Huang, Yu Li, Keyu Zhao, Weiquan Lin, Yining Zhang, Qingbin Zeng, Zhiyu Chen, Tianxing Li, Yifei Huang, and 1 others. 2025. Omniscientist: Toward a co-evolving ecosystem of human and ai scientists. *arXiv preprint arXiv:2511.16931*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guijin Son, Jiwoo Hong, Honglu Fan, Heejeong Nam, Hyunwoo Ko, Seungwon Lim, Jinyeop Song, Jinha Choi, Gonçalo Paulo, Youngjae Yu, and 1 others. 2025. When ai co-scientists fail: Spot-a benchmark for automated verification of scientific research. *arXiv preprint arXiv:2505.11855*.

Pawin Taechoyotin and Daniel Acuna. 2025. Remor: Automated peer review generation with llm reasoning and multi-objective reinforcement learning. *arXiv preprint arXiv:2505.11718*.

Nitya Thakkar, Mert Yuksekgonul, Jake Silberg, Animesh Garg, Nanyun Peng, Fei Sha, Rose Yu, Carl Vondrick, and James Zou. 2025. Can llm feedback enhance review quality? a randomized study of 20k reviews at iclr 2025. *arXiv preprint arXiv:2504.09737*.

Songjun Tu, Jiahao Lin, Qichao Zhang, Xiangyu Tian, Linjing Li, Xiangyuan Lan, and Dongbin Zhao. 2025a. Learning when to think: Shaping adaptive reasoning in r1-style models via multi-stage rl. *Neural Information Processing Systems (NeurIPS)*.

Songjun Tu, Qichao Zhang, Jingbo Sun, Yuqian Fu, Linjing Li, Xiangyuan Lan, Dongmei Jiang, Yaowei Wang, and Dongbin Zhao. 2025b. Perception-consistency multimodal large language models reasoning via caption-regularized policy optimization. *arXiv preprint arXiv:2509.21854*.

Yixuan Weng, Minjun Zhu, Guangsheng Bao, Hongbo Zhang, Jindong Wang, Yue Zhang, and Linyi Yang. 2025a. Cycleresearcher: Improving automated research via automated review. In *The Thirteenth International Conference on Learning Representations (ICLR)*.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. 2025b. Deepscientist: Advancing frontier-pushing scientific findings progressively. *arXiv preprint arXiv:2509.26603*.

Wenqing Wu, Haixu Xi, and Chengzhi Zhang. 2024. Are the confidence scores of reviewers consistent with the review content? evidence from top conference proceedings in ai. *Scientometrics*, 129(7):4109–4135.

Sarina Xi, Vishisht Rao, Justin Payan, and Nihar B Shah. 2025. Flaws: A benchmark for error identification and localization in scientific papers. *arXiv preprint arXiv:2511.21843*.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. 2025. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*.

Sihang Zeng, Kai Tian, Kaiyan Zhang, Yuru Wang, Junqi Gao, Runze Liu, Sa Yang, Jingxuan Li, Xinwei Long, Jiaheng Ma, and 1 others. 2025. Reviewrl: Towards automated scientific review with rl. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 16942–16954.

Haoxuan Zhang, Ruochi Li, Sarthak Shrestha, Shree Harshini Mamidala, Revanth Putta, Arka Krishan Aggarwal, Ting Xiao, Junhua Ding, and Haihua Chen. 2025a. Reviewguard: Enhancing deficient

peer review detection via llm-driven data augmentation. *arXiv preprint arXiv:2510.16549*.

Tianmai M Zhang and Neil F Abernethy. 2025. Reviewing scientific papers for critical problems with reasoning llms: Baseline approaches and automatic evaluation. *arXiv preprint arXiv:2505.23824*.

Yunxiang Zhang, Muhammad Khalifa, Shitanshu Bhushan, Grant D Murphy, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2025b. Mlrc-bench: Can language agents solve machine learning research challenges? *arXiv preprint arXiv:2504.09702*.

Ruiyang Zhou, Lu Chen, and Kai Yu. 2024. Is llm a reliable reviewer? a comprehensive evaluation of llm on automatic paper reviewing tasks. In *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (LREC-COLING 2024)*, pages 9340–9351.

Changjia Zhu, Junjie Xiong, Renkai Ma, Zhicong Lu, Yao Liu, and Lingyao Li. 2025a. When your reviewer is an llm: Biases, divergence, and prompt injection risks in peer review. *arXiv preprint arXiv:2509.09912*.

Minjun Zhu, Yixuan Weng, Linyi Yang, and Yue Zhang. 2025b. Deepreview: Improving llm-based paper review with human-like deep thinking process. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, page 29330–29355.

A Related Works Details

AI-assisted Scientific Research Recent advances have enabled AI-assisted scientific research across the research lifecycle, including idea formulation, experimentation, and paper writing (Ren et al., 2025; Chen et al., 2025b; Schmidgall et al., 2025). This line of work includes end-to-end research automation systems such as *The AI Scientist* (Lu et al., 2024) and its extensions (Yamada et al., 2025), as well as specialized scientific agents for problem solving and researcher training (Weng et al., 2025b; Shao et al., 2025). Scientific idea generation has also been explored through chaining strategies (Li et al., 2025b), self-reflection-guided beam search (Hu et al., 2025), and Monte Carlo tree search (Garikaparthi et al., 2025), alongside benchmarks for evaluating idea diversity and quality (Ruan et al., 2024; Qiu et al., 2025). Despite this progress, autonomously conducting complex scientific experiments remains challenging for current systems (Chen et al., 2025a; Kon et al., 2025; Zhang et al., 2025b), whereas LLMs have demonstrated strong capability in drafting academic papers when provided with existing materials and results (Ren and Wang, 2025; Hou et al., 2025). Complementary to AI research systems, we focus on paper review, emphasizing critical assessment and error-aware feedback for refinement

AI-based Paper Review Recent work on automated peer review has been framed within broader efforts to advance automated research workflows (Weng et al., 2025a), training LLMs via SFT and RL for human-like deep reasoning (Zhu et al., 2025b; Taechoyotin and Acuna, 2025) and for generating comprehensive, factually grounded reviews (Zeng et al., 2025). However, accumulating evidence shows that current AI reviewers have limited critical understanding, often missing substantive weaknesses in scientific papers (Wu et al., 2024; Zhou et al., 2024; Li et al., 2025c). They are also susceptible to perturbations in papers or rebuttals (Li et al., 2025a) and frequently fail to detect flawed reasoning, leading to biased or unreliable review judgments (Dycke and Gurevych, 2025). To address these limitations, some studies evaluate LLMs on papers with author-acknowledged errors from *Withdrawn arXiv* and *PubPeer* (Son et al., 2025; Zhang and Abernethy, 2025), while parallel efforts improve review reliability via AI feedback (Thakkar et al., 2025) or synthetic-data-based detection of deficient peer reviews (Zhang et al., 2025a). In contrast, we introduce a comprehensive dataset with accompanying error detection workflows that span multiple error types, supporting more critical peer review under perturbations.

Comparing to Existing Benchmarks Existing benchmarks for paper review and error analysis differ substantially along data provenance, error granularity, task formulation, and workflow design, as summarized in Table 1. SPOT (Son et al., 2025) and PRISMM-Bench (Selch et al., 2025) are constructed from real review ecosystems, drawing author-confirmed errors from Withdrawn arXiv and PubPeer or reviewer-flagged multimodal inconsistencies from OpenReview. While both operate at the document level, their scale is limited and supervision is narrow: SPOT evaluates error existence verification, and PRISMM formulates cross-modal inconsistencies as structured QA tasks, without modeling distributed or heterogeneous errors across a paper. FLAWS (Xi et al., 2025) adopts a synthetic setting by injecting a single claim-invalidating flaw into each paper, producing paper–error pairs with explicit span-level localization supervision. In contrast, Aspect-Guided Multi-Level Perturbation Analysis (Li et al., 2025a) is not an error annotation benchmark but a robustness analysis framework, applying controlled perturbations to papers, reviews, and rebuttals to study reviewer sensitivity, without providing error labels or detection targets. Overall, prior resources either rely on scarce real error cases, focus on single-error or localized verification, or analyze review robustness without explicit error supervision. PaperAuditBench complements these efforts by introducing controlled, document-level errors spanning multiple categories and distributed across full papers, together with multi-stage detection and explicit integration into automated review workflows.

B Additional Evaluation Details

B.1 Fast-Mode Detection Performance Breakdown

Table 5 provides a comprehensive breakdown of detection performance under the *Fast* mode across detection models and synthesis models. Each entry reports Macro-F1, Error Coverage (EC), and Finding Precision (FP), highlighting distinct detection behaviors across models. Stronger models such as GPT-5 and Gemini-2.5-Pro achieve higher overall effectiveness, while lighter or more conservative models tend to trade coverage for precision

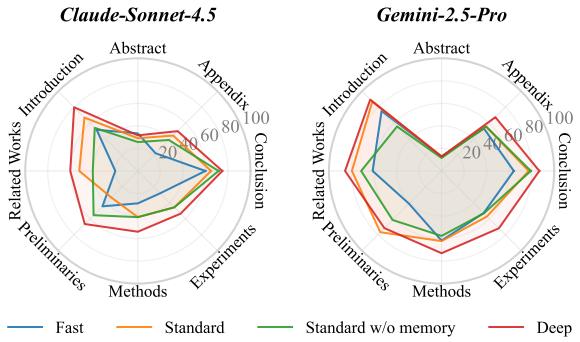


Figure 7: Error coverage across paper sections under different detection settings.

by producing fewer findings. This detailed analysis complements the aggregate trends discussed in the main text.

B.2 Additional Detectability Analysis

Detectability by Error Type Table 6 reports error coverage across different corruption types under various detection settings. Across all model combinations, increasing analytical depth consistently improves coverage for most error categories. However, detectability varies substantially by error type. Errors involving exaggerated claims or rhetorical manipulation, such as Claim Distortion (CD) and Rhetoric Bias (RB), exhibit systematically lower coverage, reflecting the difficulty of identifying subtle over-claiming without explicit contradictions. In contrast, Context Misalignment (CM), Method Logic (ML), and Ethical Omission (EO) achieve higher coverage, as these errors often involve concrete inconsistencies, invalid reasoning steps, or missing disclosures. Notably, removing global memory leads to a consistent drop in coverage, particularly for cross-section consistency errors (CS*), underscoring the role of document-level context in reliable error detection.

Detectability by Paper Section Figure 7 illustrates error coverage across different paper sections under varying detection settings. Errors located in the Abstract are consistently the hardest to detect across models and depths. This observation aligns with human reviewing practice: abstracts are typically read early and shape initial impressions, making subtle abstract-level issues more likely to be overlooked before detailed inspection of the full paper.

B.3 Metric for Human-Agent Alignment

This appendix provides detailed definitions of the ranking-based metrics used to evaluate alignment between automated reviewers and human judgments, complementing the summarized results reported in Section 5.2.

Human and Model Ranking Keys. For each paper, human judgment is represented by a ranking key $k_h = (\mu, -\sigma^2)$, where μ is the mean overall score across all human reviewers and σ^2 is the sample variance. When mean scores are tied, papers with lower inter-reviewer variance are ranked higher, reflecting greater reviewer consensus.

For automated reviewers, we construct a lexicographic ranking key: $k_m = (\textit{overall}, \textit{novelty}, \textit{technical_quality}, \textit{clarity})$, where higher values indicate better assessments. When overall scores are tied, auxiliary dimensions are used sequentially to break ties. Missing sub-scores are treated as $-\infty$, ensuring they do not win tie-breaks.

Percentile Rank Normalization. Given a set of N papers with ranking keys $\{k_i\}_{i=1}^{N}$, we first sort them in ascending order and assign average ranks to tied items. Each paper is then mapped to a percentile rank $r_i = \frac{\text{rank}(k_i)}{N-1} \in [0, 1]$, where higher values correspond to better-ranked papers. This normalization ensures comparability across datasets and avoids assumptions about score scales.

Rank-MSE. Rank Mean Squared Error (Rank-MSE) measures the squared deviation between human and model percentile ranks:

$$\text{Rank-MSE} = \frac{1}{N} \sum_{i=1}^{N} (r_i^{(m)} - r_i^{(h)})^2,$$

where $r_i^{(m)}$ and $r_i^{(h)}$ denote the model and human percentile ranks, respectively. Lower values indicate closer alignment between rankings.

Spearman's ρ. Spearman's rank correlation coefficient ρ is computed as the Pearson correlation between human and model percentile ranks:

$$\rho = \text{corr}\left(r^{(m)}, r^{(h)}\right).$$

This metric captures global monotonic consistency between rankings and is insensitive to absolute score differences.

Detect \ Synth	GPT-5	O4-Mini	Gemini-2.5	Claude-4.5
Gemini-2.5-Pro	60.5 / 57.9 / 64.1	44.6 / 51.3 / 40.2	39.7 / 38.5 / 42.3	35.4 / 35.5 / 36.5
GPT-5	62.5 / 72.4 / 56.0	42.1 / 60.8 / 33.6	37.4 / 44.7 / 33.5	38.7 / 46.5 / 34.5
Grok-4	48.1 / 39.4 / 66.3	45.2 / 43.4 / 50.1	31.6 / 25.8 / 43.3	35.1 / 29.4 / 43.6
Claude-4.5	43.1 / 39.4 / 49.0	36.1 / 39.2 / 34.6	29.9 / 28.5 / 33.1	47.3 / 46.3 / 50.2
Qwen3*	45.7 / 43.3 / 49.6	40.5 / 46.2 / 36.9	27.6 / 26.5 / 29.9	36.2 / 36.8 / 38.4
O4-Mini	45.3 / 45.0 / 48.9	39.2 / 46.9 / 35.5	27.8 / 27.8 / 29.8	30.7 / 33.7 / 32.2
GLM-4.6	35.5 / 31.5 / 42.4	38.7 / 39.9 / 39.8	26.1 / 23.1 / 32.1	38.1 / 37.3 / 42.6
Doubao-1.6	37.9 / 26.5 / 70.2	35.4 / 26.3 / 59.3	23.5 / 16.7 / 49.3	27.5 / 20.4 / 47.1
DeepSeek-V3.1*	43.6 / 37.2 / 53.6	23.2 / 22.4 / 26.8	15.9 / 12.8 / 24.3	26.2 / 22.2 / 36.1
Kimi-K2*	21.6 / 16.0 / 36.6	14.4 / 12.1 / 19.8	18.2 / 14.0 / 28.0	27.4 / 22.5 / 39.2
Detect \ Synth	**Grok-4**	**Qwen3-VL**	**Doubao-1.6**	**GLM-4.6**
Gemini-2.5-Pro	34.9 / 34.2 / 36.9	38.4 / 35.7 / 43.7	47.6 / 58.1 / 41.1	30.3 / 30.8 / 30.9
GPT-5	33.8 / 41.0 / 30.4	38.1 / 43.7 / 35.7	42.9 / 65.4 / 33.0	29.1 / 37.1 / 25.0
Grok-4	28.6 / 24.3 / 37.9	31.0 / 23.5 / 50.4	47.0 / 46.4 / 50.3	23.2 / 19.6 / 33.3
Claude-4.5	33.9 / 31.4 / 39.3	29.6 / 26.6 / 35.4	33.2 / 37.2 / 31.3	27.2 / 27.6 / 28.8
Qwen3*	29.4 / 29.2 / 31.8	31.2 / 29.3 / 34.4	35.7 / 43.7 / 31.1	25.1 / 26.1 / 25.2
O4-Mini	27.7 / 28.6 / 29.7	27.4 / 25.9 / 30.6	37.8 / 48.5 / 32.8	23.2 / 24.7 / 23.4
GLM-4.6	29.6 / 28.8 / 31.5	27.0 / 24.0 / 33.1	33.6 / 36.8 / 31.9	25.3 / 23.8 / 28.0
Doubao-1.6	21.7 / 15.1 / 43.0	19.4 / 13.1 / 42.3	42.0 / 32.8 / 62.6	13.3 / 9.5 / 24.6
DeepSeek-V3.1*	19.2 / 15.2 / 30.9	21.7 / 17.1 / 33.4	20.2 / 20.3 / 21.0	19.0 / 16.8 / 24.1
Kimi-K2*	20.9 / 15.8 / 33.1	15.4 / 11.1 / 27.4	16.4 / 14.6 / 19.9	15.9 / 12.9 / 23.0

Table 5: Fast-mode detection performance by detection and synthesis models. Each entry reports Macro-F1 / Error Coverage (EC) / Finding Precision (FP) (%). Models marked with * denote non-multimodal settings.

Kendall's τ-b. Kendall's τ-b evaluates pairwise ordering agreement while explicitly accounting for ties. Let C and D denote the numbers of concordant and discordant paper pairs, and let T_h and T_m denote pairs tied only in human or model rankings, respectively. The τ-b coefficient is defined as

$$\tau_b = \frac{C - D}{\sqrt{(C + D + T_h)(C + D + T_m)}}.$$

Values range from -1 (complete disagreement) to 1 (perfect agreement), with 0 indicating no rank correlation.

Pairwise Accuracy. Pairwise Accuracy measures the proportion of concordant pairs among all comparable (non-tied) paper pairs:

$$\text{Pairwise Accuracy} = \frac{C}{C + D}.$$

This metric has an intuitive interpretation as the probability that the automated reviewer orders a randomly chosen pair of papers in the same direction as human reviewers.

Interpretation. Together, these metrics provide complementary perspectives on alignment. Rank-MSE emphasizes absolute deviations in normalized rank positions, Spearman's ρ captures global monotonic trends, Kendall's τ-b focuses on pairwise consistency under ties, and Pairwise Accuracy offers an interpretable agreement rate. All metrics operate purely on relative rankings, avoiding reliance on absolute score calibration, which is known to vary substantially across both human and automated reviewers.

B.4 Coverage-based Alignment Protocol

For coverage-based alignment, we collect real human reviews from ICLR 2026 submissions and first consolidate multiple reviewer comments into a single unified human review summary. This consolidated summary serves as the reference representation of human judgment. The AI-generated review is then compared against this human summary using the coverage-based alignment prompt shown in Table 29.

To ensure a fair comparison, when an automated reviewer (e.g., *DeepReview*) produces multiple reviewer-style outputs, we apply the same consolidation procedure to merge them into a single unified AI review before evaluation. This normalization step ensures that coverage alignment reflects differences in *content coverage* rather than artifacts

Detect	Synth	Mode	Overall	CD	CM	EO	EM	EP	ML	RF	RB	CS*
Claude-Sonnet-4.5	***O4-Mini***	Fast	35.1	30.2	56.1	37.1	28.6	25.3	34.0	32.4	46.2	53.6
		Standard	48.4	44.8	62.1	50.0	40.3	37.3	46.2	58.1	52.6	56.5
		+ *w/o memory*	45.8	40.6	47.0	54.8	40.3	46.7	45.3	50.5	46.2	46.4
		Deep	57.3	44.8	72.7	59.7	48.7	46.7	61.3	67.6	61.5	65.2
Gemini-2.5-Pro	***GPT-5***	Fast	53.8	48.9	51.6	56.2	54.4	54.5	58.3	57.4	42.0	50.0
		Standard	60.7	51.1	60.4	67.2	60.8	54.5	66.5	69.3	55.1	58.3
		+ *w/o memory*	55.4	42.3	51.6	67.2	52.6	50.3	62.4	70.3	49.3	51.4
		Deep	70.3	62.0	67.0	78.1	69.0	70.6	74.3	80.2	59.4	64.7
GPT-5	***GPT-5***	Fast	67.7	59.1	64.8	60.9	69.0	76.2	78.0	65.3	44.9	64.0
		Standard	69.6	54.0	65.9	82.8	63.2	65.7	82.6	84.2	55.1	66.5
		+ *w/o memory*	65.3	50.4	62.6	78.1	62.0	60.1	73.9	82.2	53.6	61.2
		Deep	80.6	62.0	75.8	89.1	78.4	86.0	89.4	93.1	63.8	73.7

Table 6: Error Coverage (%) across different error types under various detection settings.

of reviewer multiplicity.

Coverage-based Alignment Metrics. Let $\mathcal{S}_H$ and $\mathcal{W}_H$ denote the sets of strength points and weakness points extracted from the human review, and $\mathcal{S}_A$ and $\mathcal{W}_A$ denote the corresponding sets extracted from the AI review. Each point is treated as an atomic, de-duplicated semantic unit.

Strength Coverage Recall (Human $\rightarrow$ AI). This metric measures the extent to which the AI review covers the strengths identified by human reviewers:

$$\text{Str-Cov} = \frac{|\mathcal{S}_H \cap \mathcal{S}_A|}{|\mathcal{S}_H|}.$$

It reflects how well the AI reviewer captures the core contributions and positive aspects emphasized by humans.

Weakness Coverage Recall (Human $\rightarrow$ AI). This metric measures the extent to which the AI review covers the weaknesses or concerns raised by human reviewers:

$$\text{Weak-Cov} = \frac{|\mathcal{W}_H \cap \mathcal{W}_A|}{|\mathcal{W}_H|}.$$

It captures the AI reviewer's ability to attend to the same limitations, methodological issues, or missing analyses highlighted by humans.

AI Extra Major Points Rate. This metric quantifies the proportion of major points introduced by the AI review that are not mentioned in the human review. Let $\mathcal{M}_A = \mathcal{S}_A \cup \mathcal{W}_A$ denote the set of major points emphasized by the AI reviewer. The metric is defined as:

$$\text{AI-Extra} = \frac{|\mathcal{M}_A \setminus (\mathcal{S}_H \cup \mathcal{W}_H)|}{|\mathcal{M}_A|}.$$

Additional points are not necessarily incorrect; rather, this metric reflects divergence in coverage scope. Lower values indicate closer alignment with human-prioritized issues.

Symmetric Coverage Similarity. This metric provides a holistic, symmetric measure of content-level agreement between human and AI reviews. Let $\mathcal{U} = \mathcal{S}_H \cup \mathcal{W}_H \cup \mathcal{S}_A \cup \mathcal{W}_A$ denote the union of all points, and $\mathcal{I} = (\mathcal{S}_H \cap \mathcal{S}_A) \cup (\mathcal{W}_H \cap \mathcal{W}_A)$ denote the matched points. The symmetric coverage similarity is computed as:

$$\text{Sym-Cov} = \frac{|\mathcal{I}|}{|\mathcal{U}|}.$$

This metric summarizes overall overlap while penalizing both missed human points and excessive divergence introduced by the AI review.

B.5 Error Categories Associated with Review Score Drops

To identify which detected issues most strongly contribute to numerical score reductions, we conduct a regression-style analysis that relates changes in review scores to newly surfaced or intensified critiques categorized by PaperAudit error types. For each paper, we compare baseline and final reviews produced by GPT-5 and examine how the presence of different error categories correlates with subsequent score decreases. Table 7 summarizes the results under two complementary settings: GPT-5 reviews on ICML 2025 papers with synthetically injected errors (top), and real GPT-5 reviews of ICLR 2026 submissions (bottom). For each error category, we report the number of score drops, the empirical drop rate, and a relative *lift*, defined as the ratio between the probability of observing that

Error Type	#Drop	Drop Rate	Lift	#Total
Method Logic Errors	9	0.26	1.52	35
Context Misalignment	8	0.22	1.25	36
Claim Distortion	5	0.19	1.00	27
Rhetoric Bias	2	0.10	0.46	21
Experiment Protocol Flaws	1	0.08	0.40	12
Evidence Manipulation	1	0.12	0.63	8

(a) **ICML 2025 (synthetic errors with GPT-5).**

Error Type	#Drop	Drop Rate	Lift	#Total
Claim Distortion	5	0.33	1.88	15
Method Logic Errors	11	0.31	1.65	36
Context Misalignment	4	0.15	0.68	26
Rhetoric Bias	3	0.16	0.71	19
Experiment Protocol Flaws	2	0.14	0.63	14
Evidence Manipulation	0	0.00	0.00	9

(b) **ICLR 2026 (real submissions with GPT-5 reviewers).**

Table 7: **Error categories most strongly associated with review score drops.** Top: analysis on GPT-5–synthesized errors for ICML 2025 papers. Bottom: regression analysis on real GPT-5 reviews for ICLR 2026 submissions. Lift > 1 indicates a higher likelihood of score reduction relative to non-drop cases.

error in score-decreasing reviews and its probability in non-decreasing reviews. A lift value greater than one therefore indicates that the error type is more strongly associated with score reductions.

Across both settings, error categories that directly affect technical soundness, most notably *method logic errors* and *claim or result distortion*, consistently exhibit higher drop rates and lift values, indicating a stronger influence on scoring decisions. In contrast, issues related to rhetorical presentation, experimental protocol details, or evidence handling occur frequently but show substantially weaker associations with numerical score decreases. Overall, this analysis suggests that review score reductions are driven not by the sheer number of detected issues, but by whether those issues undermine methodological consistency or the credibility of reported claims, reinforcing our main conclusion that PaperAudit-Review enables more discriminative and substantively grounded reviewing behavior.

C Detailed Pipeline of PaperAudit-Bench

C.1 Origin Data Preprocessing

We provide *implementation-level details* of the preprocessing pipeline used to transform raw conference paper PDFs into the structured, section-aware JSON representations described in Section 3.1. We focus on the concrete engineering steps for parsing, cleaning, and organizing original papers prior to synthetic error injection and auditing.

Paper Acquisition and Length Filtering. Source papers are collected from OpenReview through official conference APIs. For each accepted paper, we create an isolated directory containing the PDF file and associated metadata. This directory-level isolation is preserved throughout the pipeline to support independent parsing, failure recovery, and reliable alignment between original and corrupted versions. Before parsing, we apply a lightweight page-count filter to exclude overly long PDFs, which empirically lead to unstable OCR or layout extraction due to large appendices or image-heavy content. Rather than truncating documents, such cases are discarded to ensure consistent and reliable downstream processing.

Multimodal Parsing and Content Extraction. Following (Son et al., 2025), each retained PDF is converted into a structured multimodal representation using LlamaParse. Text is extracted at the page level and refined through a second-stage OCR correction step, where a language model repairs common layout artifacts (e.g., broken formulas, hyphenation, and line wrapping) while preserving mathematical expressions and LaTeX-style notation. In parallel, visual elements such as figures and tables are extracted. Redundant sub-images and page-level screenshots are removed via bounding-box containment checks and template-based matching, and remaining candidates are filtered using a lightweight classifier to distinguish manuscript figures from non-content artifacts. Validated visual blocks are then interleaved with text blocks to form a unified content sequence.

Section Labeling and Structural Refinement. To support section-aware reasoning, parsed text blocks are normalized and segmented with respect to semantic section boundaries, avoiding arbitrary fragmentation within the same section. Each content block (text or figure) is assigned a global index and a section label from a fixed predefined set. Section labels are represented as contiguous index ranges to guarantee full document coverage. Figures and tables are labeled independently based on captions and semantic cues, allowing their assigned section to differ from surrounding text when appropriate.

https://cloud.llamaindex.ai/

Rule-based Cleanup and Final Representation. We apply additional rule-based post-processing to remove systematic noise irrelevant to paper auditing. Repeated page headers and footers are detected via frequency analysis across blocks and removed while preserving their first occurrence. Conference checklist sections are also discarded, as they are orthogonal to scientific claims and methodological reasoning. After cleanup, block indices are re-normalized and section spans are rebuilt to maintain internal consistency.

The final output is a section-aware JSON representation in which each paper is stored as an ordered sequence of indexed multimodal content blocks, serving as the standardized input for error injection, detection, and review experiments. Table 8 provides an illustrative snippet of the resulting representation.

C.2 Synthetic Error Injection

This subsection summarizes how we inject synthetic errors on top of the processed, section-aware JSON representation. Concretely, we query a LLM to generate a set of edit patches under strict JSON constraints (Table 16), and then deterministically apply these patches to the processed paper content. Each injected corruption is assigned to one of the eight error categories in Table 9.

Edit-based Corruption as JSON Patches. We model corruption as a set of *edit patches* applied to the processed paper content. Given the paper represented as an ordered list of multimodal blocks, the LLM is prompted to propose 10–20 localized edits, where each edit specifies (i) a verbatim text span to locate (`target_find`) and (ii) its replacement (`replacement`), together with lightweight metadata such as the error category ({Error_Type}), the target section, and a short reviewer-style explanation (Table 16). For each modification, we maintain a complete and traceable record, including the original text span, the edited version, the associated category, and a natural-language explanation. To ensure that edits are precisely grounded and reproducible, `target_find` is required to be copied character-for-character from the original text blocks; we then apply the corruption by performing a single-span replacement at the first matched occurrence. We generate patches with broad section coverage (at least four sections) so that injected errors are not concentrated in a single region, better reflecting diverse failure modes that may appear in real reviewing.

Section-aware Prompting over Block Streams. The injector prompts the LLM with the paper represented as an ordered stream of multimodal blocks, each annotated with an index and section label as anchors (Table 8). Text blocks are provided verbatim (with truncation when needed), while figures and tables are included as `image_url` items (base64 `data:` URLs) to support visually grounded corruptions. This formulation avoids collapsing the entire PDF into a single long context while preserving global order and section locality, enabling controlled within- and cross-section corruptions. Each injected patch is finally labeled with an error category ({Error_Type}) from the taxonomy in Table 9, spanning evidence manipulation, method logic, protocol flaws, claim distortion, context misalignment, reference fabrication, ethical omission, and rhetoric bias.

C.3 Error Detection Pipeline

This subsection provides implementation-level and formal details of the error detection workflow summarized in Section 4.1. Given a synthesized paper represented as a section-aware block stream (Section C.2), the detector outputs a unified list of reviewer-style findings under a strict JSON schema.

The pipeline consists of three complementary detection modules corresponding to the *Fast*, *Standard*, and *Deep* modes described in the main paper: **(i) global cross-section review**, **(ii) section-level review with optional global memory**, and **(iii) task-based multi-agent review**. Each module is designed to expose different classes of issues, ranging from local within-section inconsistencies to cross-section contradictions (often required for error types such as {Error_Type} in Table 9).

Fast Mode: Global Cross-section Review. In the Fast mode, a document-level detection agent directly operates on the full paper representation P in a single pass, without explicit section decomposition or multi-step reasoning. The detection result is given by

$$\mathcal{F}_{\text{fast}} = f_{\text{det}}(P), \tag{2}$$

where $f_{\text{det}}(\cdot)$ denotes a full-document reviewer that scans the ordered block stream and outputs a set of findings. The reviewer is explicitly instructed to prioritize *cross-section support mismatches*, such as abstract-level claims not supported by experimental evidence or numerical inconsistencies across

Example: processed paper JSON representation.


```
"content": [
  {
    "type": "text",
    "text": "Mean Flows for One-step Generative Modeling\n
            Zhengyang Geng ... Kaiming He ...",
    "index": 1,
    "section": "Abstract"
  },
  {
    "type": "text",
    "text": "Abstract\nWe propose a principled and effective framework for
            one-step generative modeling ... (truncated)",
    "index": 2,
    "section": "Abstract"
  },
  {
    "type": "image_url",
    "image_url": "...
                 ... (truncated)",
    "index": 3,
    "section": "Experiments"
  },
  {
    "type": "text",
    "text": "1 Introduction\nThe goal of generative modeling is to transform
            a prior distribution into the data distribution ... (truncated)",
    "index": 4,
    "section": "Introduction"
  },
  ...
]
```


Table 8: Illustrative snippet of a processed paper representation. The `content` list interleaves section-aware text blocks with image blocks (encoded as base64 `data:` URLs), and assigns a global index to each block for precise downstream localization.

text and tables, while remaining concise and non-redundant. The detailed prompt and strict JSON output schema are provided in Table 17.

Standard Mode: Section-level Review with Global Memory. The Standard mode performs section-aware detection augmented with an explicit global memory. Given an input paper P, a global memory representation

$$m = f_{\text{mem}}(P) \tag{3}$$

is first constructed from the full document, encoding high-level information such as the problem formulation, main contributions, experimental setup, and key claims. The paper is then segmented into a set of section slices $\{s_i\}_{i=1}^{N}$, and each section s_i is independently reviewed by a section-level detector conditioned on both the local section content and the shared global memory. The resulting findings are consolidated by a merging operator:

$$\mathcal{F}_{\text{standard}} = f_{\text{merge}}\Big(\big\{f_{\text{det}}(s_i, m)\big\}_{i=1}^{N}\Big). \tag{4}$$

This design enables fine-grained within-section analysis while supporting cross-section reasoning through shared global context. The corresponding prompt template and unified finding schema are shown in Table 18.

Deep Mode: Task-based Multi-agent Review (Planner → Retriever → Specialist). The Deep mode extends the Standard setting by introducing explicit planning and specialized analysis. For each section s_i, a Planner agent generates a targeted review plan

$$\pi_i = f_{\text{plan}}(s_i, m), \tag{5}$$

which specifies the review objectives and risk dimensions to be examined. For each plan, a Retriever agent optionally gathers task-specific supporting evidence

$$e_i = f_{\text{ret}}(s_i, m, \pi_i), \tag{6}$$

including external evidence via web search when necessary. Guided by (π_i, e_i), domain-specific Specialist agents conduct focused reviews and produce

Error Type	Description
Evidence Manipulation (EM)	Incorrect or misleading use of empirical evidence, including fabricated results, altered metrics, or unsupported performance claims that do not faithfully reflect the reported data.
Method Logic Errors (ML)	Logical inconsistencies or invalid reasoning in method or algorithm descriptions, such as incorrect derivations, unjustified assumptions, or flawed step-to-step transitions.
Experiment Protocol Flaws (EP)	Errors in experimental design or evaluation protocols, including invalid baselines, improper ablations, inconsistent evaluation settings, or misleading reporting of experimental procedures.
Claim Distortion (CD)	Overstated, unsupported, or incorrect interpretations of experimental results, where conclusions do not logically follow from the presented evidence or fail under reasonable perturbations.
Context Misalignment (CM)	Inconsistencies across different sections of a paper, such as mismatches between methods, experimental setups, and reported results, often requiring cross-section reasoning to detect.
Reference Fabrication (RF)	Incorrect, misleading, or fabricated citations, including references that do not support the associated claims or formal arguments that misrepresent prior work.
Ethical Omission (EO)	Omission or incomplete disclosure of ethical considerations, such as unreported computational resources, missing risk discussions, or incomplete experimental disclosures.
Rhetoric Bias (RB)	Misleading rhetorical strategies that exaggerate novelty or effectiveness, including biased phrasing, unqualified superlatives, or claims of dominance without sufficient justification.

Table 9: Detailed descriptions of the eight error categories used in PaperAudit-Dataset.

findings grounded in the retrieved evidence and local context. The final detection results are obtained by consolidating all specialist outputs:

$$\mathcal{F}_{\text{deep}} = f_{\text{merge}}\Big(\{f_{\text{spec}}(s_i, m, \pi_i, e_i)\}_{i=1}^{N}\Big). \quad (7)$$

This task-based multi-agent design improves coverage and encourages systematic exploration of diverse error modes, at the cost of higher computational complexity. Prompt templates for the Planner, Retriever, and Specialist roles are provided in Table 19.

Findings Merge and Evaluation. Findings produced by the Standard, and Deep modes are consolidated through a conservative merge stage that removes exact or near-duplicate reports while preserving all distinct, well-supported issues. This merge step performs adjudication rather than re-detection: it does not introduce new findings or evidence, but only unifies semantically identical reports in Table 20. For evaluation, we adopt an LLM-as-a-Judge protocol to match merged findings against injected ground-truth (GT) errors. A GT error is considered detected if any finding captures the same underlying issue, allowing paraphrasing and partial span overlap. Matching is performed conservatively and is not based on exact string matching. The evaluation prompt and matching criteria are provided in Table 21.

C.4 Training Details of Lightweight Detectors

Hyperparameters Table 10 summarizes the hyperparameters used for SFT and RL, as well as the LoRA configuration and optimization settings.

Settings	Hyperparameters
Training (SFT)	train_batch_size = 2 learning_rate = 5e-5 epochs = 3.0 warmup_ratio = 0.1 scheduler = cosine
Training (RL)	rollout_batch_size = 2 train_batch_size = 64 micro_batch_size_per_gpu = 1 epochs = 3 learning_rate = 5e-5 scheduler = cosine
LoRA	r = 64 lora_alpha = 128 lora_dropout = 0.05 target_modules = all bias = none task_type = CAUSAL_LM
Optimization	flash_attn = true adam_offload = true deepspeed = ZeRO-3

Table 10: Hyperparameters used in the training of Error Detection Model.

Training Cost. For the SFT phase, all experiments were conducted using 4 NVIDIA H100 GPUs: the training of Llama 3.2-3B completed in 0.5 hours, Qwen3-8B took 2 hours, and Qwen3-14B required 3 hours. During the subsequent RL training phase, the Qwen3-80B model was deployed as the reward model on a server with 4 NVIDIA H100 GPUs, which was accessed via port calls. RL training itself utilized 8 NVIDIA H100 GPUs: Llama 3.2-3B finished in 2 hours, Qwen3-8B took 4 hours, and Qwen3-14B required 7 hours.

Case 1 (A Training Instance for Lightweight Detectors)

[Input]
You are an expert academic paper reviewer. Your task is to carefully analyze the following text from a research paper and identify *ALL* potential scientific errors, methodological flaws, or integrity issues.
Important: The text below may contain multiple errors. You must detect and report each error independently.
Text to Review:
KV Cache Compression. Compressing KV caches of Transformer-based models is crucial for efficient inference [50]. ..... KVzip builds on groundbreaking work in [61], which first proposed query-agnostic eviction, making our method a direct extension. Unlike existing methods that perform query-dependent KV compression, we propose query-agnostic compression, enabling compressed .....
Instructions:
Analyze the text above and identify all errors. For each error found, provide:
- `error_location`: Exact erroneous content for correction
- `error_type`: Type of error (evidence_data_integrity | method_logic_consistency ...)
- `error_explanation`: Detailed explanation of the error and its impact

Output Format: Return a JSON array containing all detected errors.
[Output]
```
[
 {
   "error_location": "DuoAttention profiles and selectively replaces attention heads with sliding-window attention prior to deployment [52, 53]. KVzip builds on groundbreaking work in [61], which first proposed query-agnostic eviction, making our method a direct extension.",
   "error_type": "reference_background_fabrication",
    "error_explanation": "KVzip is claimed to 'build on groundbreaking work in [61]', while [61] is cited as a retrieval-augmented generation paper, not a KV cache compression paper. This misattribution inflates novelty by falsely positioning KVzip as an extension of unrelated prior work."
 }
]
```
[Ground Truth]
```
[
 {
    "error_location": "KVzip builds on groundbreaking work in [61], which first proposed query-agnostic eviction, making our method a direct extension.",
   "error_type": "reference_background_fabrication",
   "error_explanation": "Misattributes ideas by citing a fabricated reference [61] for advanced KV compression, which does not exist and inflates novelty claims. This distorts the background, making KVzip appear more innovative than it is vs. actual priors like [33] and [6]."
 }
]
```
Performance Verification: The model achieves a matching score of **0.88** and an accuracy (ACC) of **0.88**.

Table 11: A representative training instance for the academic paper error detection task. The example specifically targets a *reference background fabrication* error (i.e., misattributing KVzip to unrelated prior work via a fabricated/cited-in-error reference [61]), demonstrating the model's capability to identify such integrity issues.

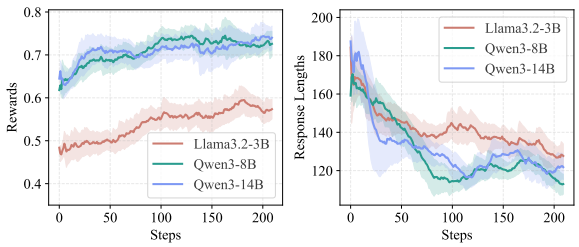


Figure 8: RL training dynamics for lightweight models.

C.5 Review Workflow Details

RL Training Dynamics. Figure 8 shows RL increases reward while reducing response length, reflecting more accurate and concise detection behavior.

Supplementary Templates and Data Examples. We present core prompts and training examples for our error detection task:

- **Table 22** defines the prompt template for the reward judge model, which evaluates detection results via two criteria (*Precision* and *Conciseness*) with weighted scoring rules.
- **Table 23** provides the SFT input prompt template, specifying the task description, error type categories, JSON output format, and a structural example.
- **Table 11** shows a representative RL training example, including the model input, output,

and reward model evaluations, which helps elucidate the data structure and the RL judgment process.

Following the workflow in Section 4.3, we examine how explicit error detection supports more critical peer review. Observed score differences arise from conservative, prompt-constrained refinements grounded in the manuscript, rather than from free-form re-evaluation.

Baseline Review and Document Memory. We first build a high-density *paper memory* from the full manuscript, which serves as a compact, section-aligned recall artifact for downstream analysis (Table 24). This memory is strictly descriptive: it preserves the paper's stated problem, method, experimental setup, key quantitative results, and explicitly reported limitations, while forbidding critique or extrapolation. Independently, a baseline reviewer then produces a standard reviewer-style assessment under a fixed structure and evidence-first constraints (Table 25), without access to any auxiliary analyses.

Auxiliary Reports as Structured Analytical Signals. Next, we construct two auxiliary reports that function as *silent attention cues* for later refinement, rather than as evidence sources. **(i) Error/Integrity Report** is generated by an integrity-focused reviewer that scans the manuscript (and optionally section slices with global memory) to surface *high-impact, clearly observable* issues (Table 26). Concretely, it prioritizes: (a) cross-section contradictions (claims vs. methods vs. results), (b) numerical or factual mismatches across text, figures, tables, and equations, and (c) missing or conflicting protocol details that prevent verification. The report is constrained to be *evidence-first* and conservative: every issue must be supported by explicit manuscript anchors (e.g., section/figure/table/equation/page), speculative accusations are prohibited, and minor stylistic issues are ignored; when direct support is unavailable, the report must explicitly state that no evidence was found. **(ii) Motivation/Novelty Report** is generated by a Senior-AC-style evaluator that cross-examines the authors' stated motivation and novelty claims against a retrieved set of similar works (Table 27). This module first extracts the claimed problem, gap, and solution from the manuscript, then evaluates whether the paper explicitly differentiates itself from similar prior works and whether such differentiation is technically substantive. Crucially, the evaluator is forbidden from hallucinating missing citations or external facts: novelty judgments must cite manuscript sections (e.g., Introduction/Related Work/Discussion) and provide a synthesized verdict (Incremental vs. Application-Oriented vs. Substantive) that goes beyond merely restating similarity evidence.

Prompt-constrained Refinement. Finally, the baseline review is refined under strict minimal-edit constraints (Table 28). The refinement prompt enforces that the baseline review remains the primary artifact: global rewriting, stylistic polishing, and speculative extensions are disallowed. Although the two auxiliary reports are provided to guide where the reviewer should re-check the manuscript, the reviewer is explicitly forbidden from citing, quoting, or referring to them in any form. All critiques and score adjustments must be justified exclusively by manuscript evidence with explicit anchors; if no direct evidence can be identified, concerns are not escalated. Together, these constraints ensure that structured analyses calibrate the review process without replacing it, yielding stricter yet accountable human-style peer review.

D Case Study

In this section, we present three representative case studies to qualitatively illustrate the behavior of our system. In addition to the training examples provided in Table 11 related to Appendix C.4, we further showcase two end-to-end examples drawn from different stages of the pipeline: one focusing on the *error detection agent* and the other on the *review refinement agent*.

D.1 Error Detection Case

Detect Example on a Corrupted Paper. We present a case study using GPT-5 as both the corruption generator and the detection agent on a corrupted ICML-25 paper (*STAIR: Improving Safety Alignment with Introspective Reasoning*). As shown in Table 12, the three detection modes exhibit complementary behaviors. The *Fast* mode reliably captures salient surface-level inconsistencies, such as exaggerated claims and narrative–table mismatches that require minimal contextual reasoning. Errors that depend on cross-section reconciliation or theoretical consistency, including train–test contamination, normalization mismatches, and decoding protocol inconsistencies, are primarily

identified under the *Standard* and *Deep* modes. In contrast, errors that are fundamentally unverifiable from the manuscript alone, such as misreported computation costs or unexplained numerical changes, remain challenging across all modes, reflecting inherent limitations of manuscript-based auditing.

Fine-grained and Over-analysis. Deeper detection modes uncover a broader set of fine-grained error signals, but also introduce over-correction on details peripheral to the core scientific claims. As shown in Table 13, many unmatched findings correspond to valid but out-of-scope analyses, including evaluator coupling, comparability controls, and undocumented evaluation discrepancies. At the same time, Deep inspection produces over-analytical observations, such as exhaustive reproducibility checklists or presentation-level critiques, which are more appropriately viewed as reviewer-style suggestions than concrete errors. These results underscore the role of human judgment in interpreting detection outputs, with deeper modes being particularly suited for self-auditing on familiar manuscripts, where authors can selectively adopt substantive corrections while disregarding low-impact over-analysis.

D.2 Review Case

Review Comparison. We present a qualitative review case study on the corrupted paper (*STAIR*) to examine how explicit error detection influences downstream peer review. Table 14 compares the baseline review with an error-aware *PaperAudit* review produced from the same manuscript. While the baseline review focuses on conceptual contributions and reported empirical gains, the *PaperAudit* review systematically evaluates internal consistency and evaluation validity. This leads to the identification of theory inconsistencies, contamination risks, and claim–evidence mismatches that are not raised in the baseline assessment. Correspondingly, the audit-oriented review assigns more conservative scores in technical quality and clarity, reflecting evidence-based scrutiny rather than stylistic strictness.

Motivation and Error Reports. In addition to review-level differences, we also present the associated motivation report and detected error report as supplementary case-study artifacts. These reports summarize the stated research motivation and enumerate evidence-anchored consistency issues identified from the same corrupted manuscript, as shown in Table 15.

E Prompts

Tables 16–28 summarize all prompt templates used in this work. For clarity and space efficiency, some prompts are lightly condensed, while preserving their original constraints and operational semantics. Complete prompt implementations and full specifications are available in the released code.

Case 2 (Detect Example: Error Coverage across Detection Modes)			
Legend. ✓= detected ✗= missed (Columns: **F**ast / **S**tandard / **D**eep)			
Injected Error (condensed GT)	**F**	**S**	**D**
E1 (Abstract). Over-claim: surpasses Claude-3.5 on all jailbreak suites with no helpfulness loss.	✓	✓	✓
E2 (Method). Inconsistent output delimiters: `<|Final|>` vs. `<|Output|>` across sections.	✓	✓	✓
E3 (Method). Theorem inconsistency: $F(0) = 1$ contradicts Appendix proof requiring $F(0) = 0$.	✓	✓	✓
E4 (Experiments). Fabricated narrative number: AlpacaEval 48.66% vs. tables ∼38.66%.	✓	✓	✓
E5 (Experiments). Train–test contamination: StrongReject prompts included in training data.	✓	✓	✓
E6 (Appendix). Ethical impact statement weakens or omits jailbreak risk discussion.	✗	✓	✓
E7 (Experiments). Fabricated benchmark claim: HarmBench-XL listed without results.	✗	✓	✓
E8 (References). Unverifiable citation (Li et al., 2025) used to justify HarmBench-XL.	✗	✗	✗
E9 (Appendix). UCB1 formula inverted ($\ln n_i/N_i$ instead of $\ln N_i/n_i$).	✓	✗	✓
E10 (Experiments). Misreported compute time (30h → 8h on A800).	✗	✗	✗
E11 (Table). Arithmetic inconsistency: Table 7 average altered without split changes.	✗	✓	✗
E12 (Conclusion). Over-generalized claim: eliminates jailbreak vulnerability without trade-offs.	✓	✗	✓
E13 (Appendix). Removal of evaluation-set decontamination description.	✗	✓	✓
E14 (Appendix). Default decoding inconsistency: beam search vs. greedy.	✗	✗	✓
E15 (Table). Unexplained StrongReject score change (0.6753 → 0.7153).	✗	✗	✗
E16 (Method+Appendix). Rating normalization mismatch: $[-1, 1]$ vs. $[0, 1]$.	✓	✓	✓

Table 12: Detect-agent case study on a corrupted *STAIR* paper. Each row lists an injected paper-level error and whether it is detected under *Fast*, *Standard*, and *Deep* modes. Fast captures prominent surface-level inconsistencies, while Standard and Deep progressively recover cross-section and theory-sensitive errors.

Case 2 (Detect Example: Fine-grained and Over-analysis)	
A. Fine-grained Analysis (Valid but Out-of-scope to Injected GT)	
Evaluator coupling	Identifies that GPT-4o is used both to generate or filter training data and as an evaluator for StrongReject, raising **potential judge-specific bias**. This is a **substantive evaluation concern**, but not an injected corruption.
Comparability control	Points out that claims of fair comparison mix **8B open models with a 32B model**, confounding **performance attribution**. This reflects experimental rigor rather than a planted error.
Cross-table inconsistency	Notes that STAIR-DPO-3 reports **different AlpacaEval scores across tables** under seemingly identical settings, suggesting **undocumented evaluation differences**. This is distinct from the injected narrative-number manipulation.
Safety disclosure	Observes that qualitative examples include **harmful prompts without sufficient redaction or safety framing** in figures, a legitimate concern about responsible presentation.
Conceptual precision	Questions the **causal framing** that rapid refusals imply vulnerability, highlighting an **imprecise conceptual leap** rather than a factual contradiction.
B. Over-analysis (Reviewer-style or Low-specificity Observations)	
Reproducibility granularity	Requests **exhaustive disclosure of hyperparameters, random seeds, and rollout schedules** beyond what is required to verify core claims.
Algorithmic underspecification	Flags **missing threshold values and stopping criteria** in the algorithm description, which affects clarity but does not constitute a concrete error.
Figure hygiene	Focuses on **missing axis labels or unclear compute scales** in plots, a presentation issue rather than a scientific inconsistency.
Bibliographic completeness	Points out **missing venue details or formatting issues** in references, which impacts polish but not validity.
Versioning detail	Requests a **specific code commit hash or release tag**, reflecting best practices rather than an error in the manuscript.

Table 13: Case study of representative *unmatched* findings produced by the Deep detection mode. **Blue** highlights fine-grained but substantive analyses beyond the injected ground-truth scope, while **orange** marks reviewer-style or low-specificity observations.

Case 3 (Review Example: Comparison of Baseline and PaperAudit Review)	
Baseline Review	**PaperAudit Review**
Overall Tone Emphasizes conceptual novelty, strong empirical gains, and clarity of the proposed SI-MCTS + step-level DPO framework.	**Overall Tone** **More Critical and Evidence-Driven Tone**. Focuses on **validity, internal consistency, and evaluation integrity**, even when core ideas are promising.
Strengths Emphasis Highlights dual-process framing, theoretically grounded reward design, and strong StrongReject improvements.	**Strengths with Qualification** Acknowledges the same strengths, but **systematically qualifies claims** by linking them to specific assumptions, evaluation setups, and boundary conditions.
Treatment of Theory Notes the reward formulation and Theorem 2.1 as principled and well-motivated.	**Theory Auditing** **Explicitly flags the inconsistency between $F(0) = 1$ in Section 2.2 and $F(0) = 0$ in Appendix B.1**, questioning correctness of the stated theorem family and its conditions.
Evaluation and Results Accepts reported gains on StrongReject, AlpacaEval, and comparisons to open/proprietary models at face value.	**Evaluation Integrity Checks** **Identifies train–test contamination risk** from StrongReject prompts used in training and **numerical inconsistencies across text and tables** (e.g., AlpacaEval 48.66% vs. 38.66%).
Use of External Models Mentions GPT-4o usage mainly as an implementation detail for data synthesis and grading.	**Evaluator Coupling and Bias** **Highlights evaluator coupling**, where GPT-4o is used both for data construction and as the StrongReject evaluator, raising concerns about bias and inflated gains.
Algorithmic Details Treats SI-MCTS description and UCB-style selection as sufficiently specified.	**Algorithmic Correctness** **Detects incorrect UCB1 formulation** and missing equation references, directly affecting the validity of Algorithm 1.
Statistical Reporting Does not question the absence of variance estimates or confidence intervals.	**Statistical and Reporting Gaps** **Notes missing variance, error bars, and sensitivity analyses**, limiting robustness of empirical conclusions.
Scope and Claims Generally accepts abstract and conclusion claims.	**Claim–Evidence Alignment** **Flags over-generalized claims** such as "surpasses every jailbreak suite" and unsupported mentions of HarmBench-XL without reported results.
Final Scores Overall: 7 Novelty: 7 Technical: 6 Clarity: 7	**Final Scores** **Overall: 6** $\downarrow$ Novelty: 7 **Technical: 5** $\downarrow$ **Clarity: 6** $\downarrow$

Table 14: Comparing the baseline review and an *PaperAudit* review on the corrupted *STAIR* paper. The audit-oriented review systematically surfaces theory inconsistencies, evaluation integrity risks, and claim–evidence misalignments that are largely unaddressed in the baseline review. **TealBlue** highlights substantively auditable issues that directly affect correctness or evaluation validity, while **Plum** denotes reviewer-level concerns that are reasonable but not strictly erroneous.

Case 3 (Review Example: Motivation and Detected Error Reports)
Innovation & Motivation Report
Targeted Problem. The paper targets safety alignment of large language models under jailbreak scenarios, emphasizing the safety–performance trade-off induced by reflexive refusals and shallow alignment strategies.
Claimed Gap. Conventional SFT/DPO/RLHF approaches are described as effective for overt harms but vulnerable to complex jailbreaks, while existing multi-objective and deliberative alignment methods are argued to be insufficient or to require large reasoning models.
Proposed Framework. The manuscript introduces a three-stage pipeline consisting of structured chain-of-thought format alignment, iterative self-improvement via Safety-Informed Monte Carlo Tree Search, and a process reward model guiding test-time Best-of-N and beam search.
Reward Design. A theoretically motivated reward family balancing helpfulness and safety is proposed and instantiated as $R(H, S) = S \cdot H + 2S$, with stated properties such as safety priority and dual monotonicity.
Novelty Characterization. The contribution is positioned as an integration of known components into a safety-focused, process-level alignment pipeline, with novelty primarily arising from engineering synthesis and targeted application.
Academic Integrity and Consistency Risk Report
Theoretical Consistency. The reward formulation exhibits inconsistent conditions on $F(0)$ between the main theorem and the appendix, leading to incompatible assumptions for the stated theoretical family and its instantiation.
Reward Scale and Implementation. Theoretical assumptions require safety scores in $[-1, 1]$ with negative values for unsafe outputs, while implementation descriptions normalize ratings to $[0, 1]$ without documenting any re-centering, undermining stated properties.
Search Algorithm Specification. The UCB1 selection rule is described with inverted parent–child visit terms relative to the standard formulation, and a referenced equation is missing, affecting algorithmic clarity and verifiability.
Formatting and Data Specification. Final-answer and reasoning-step tag formats are inconsistent across sections and appendices, introducing ambiguity in training, parsing, and evaluation procedures.
Numerical Reporting. Central metrics such as AlpacaEval winning rates are reported inconsistently across narrative text and tables, with multiple discrepancies lacking documented changes in evaluation settings.
Claim Alignment. Abstract-level claims of per-suite superiority and zero degradation in helpfulness are not uniformly supported by the reported per-benchmark results.

Table 15: Motivation and detected error reports for the corrupted ICML-25 paper *STAIR*. **Green** indicates reported strengths and motivations, while **red** indicates reported inconsistencies and risks, without additional interpretation.

Prompt Template (Synthetic Error Injection)

System Prompt

You are a strict JSON-only generator. Output a single valid JSON object and nothing else.
You are a synthetic corruption generator for academic papers.
You will receive the paper as an ordered list of blocks (each block is either a TEXT chunk or an IMAGE URL).
Read blocks in order. Do NOT reconstruct the full paper; propose realistic edits based on the provided blocks.

Error Types

Choose exactly one `corruption_type` ({Error_Type}) per edit. Error type definitions follow Table 9.

Edit Coverage Requirements

- Produce 10–20 edits.
- Cover at least four sections among: Abstract, Introduction, Related Work, Method, Experiments, Discussion, Conclusion, References.
- Keep coherence; edits must resemble genuine manuscript content.

Output Rules

Return ONE SINGLE JSON object with keys:
`"global_explanation"` (string) and `"edits"` (list of 10–20 objects).
Each edit object must contain exactly:

```
{
  "id": 1,
  "corruption_type": "{Error_Type}",
  "difficulty": "easy|medium|hard",
  "location": "SectionName",
  "rationale": "short reason",
  "error_explanation": "reviewer-style, 3-6 sentences, with anchors",
  "needs_cross_section": true|false,
  "target_find": "verbatim substring copied from original TEXT blocks",
  "replacement": "new content replacing target_find (concise)"
}
```

Hard Constraints

- `target_find` MUST be copied verbatim from the original TEXT blocks (character-for-character).
- Prefer selecting `target_find` from TEXT blocks; if image-driven, anchor via nearest caption text.
- Do NOT output the full paper or full tables; keep edits paragraph-level and self-contained.

Table 16: Prompt template used to synthesize document-level corruptions as JSON edit patches. The error type placeholder {Error_Type} refers to the taxonomy in Table 9.

Prompt Template (Fast Mode: Global Cross-Section Review)

System Prompt

You are a rigorous academic paper reviewer specializing in scientific error detection.

Your job is to read the entire manuscript (all sections) and identify substantive problems that would matter in a real peer review.

Evidence-first: Every finding MUST be grounded with explicit anchors to the manuscript content provided (section name + block index and/or quoted span).

Do not force findings: If you cannot justify an issue from the given content, do not report it.

Prefer cross-section checks: prioritize inconsistencies across sections (e.g., Abstract claims vs Experiments evidence, conflicting numbers across sections, mismatched settings).

JSON-only: Output a single valid JSON object and nothing else.

Allowed error categories

For `type`, choose **exactly one** of: {Error_Type}. Definitions follow Table 9.

Output schema (strict)

Return:

```
{
  "findings": [
    {
      "type": "{Error_Type}",
      "section_location": "SectionName",
      "error_location": "verbatim quoted span (or block index range)",
      "evidence": [
        {"block_index": 12, "quote": "short quoted span"},
        {"block_index": 35, "quote": "short quoted span"}
      ],
    "explanation": "3-6 sentences; why it is an issue and why evidence is insufficient/inconsistent",
      "confidence": 0.00,
      "proposed_fix": "1-3 concrete sentences; how to correct or clarify"
    }
  ]
}
```

Constraints: (1) `confidence` in [0,1]. (2) No duplicates / near-duplicates. (3) Keep `evidence` quotes short (avoid copying long paragraphs).

If no issues: output `{"findings": []}`.

User Prompt

You will receive a paper as an ordered list of multimodal blocks. Each block provides:

- `index`: global block id
- `section`: semantic section label
- `type`: `text` or `image_url`
- `text` or `image_url` (base64 data URL)

Read blocks in order and produce reviewer-style findings following the schema above.

Table 17: Detailed prompt template for Fast Mode (global cross-section review). Error categories use the placeholder {Error_Type} and refer to Table 9.

Prompt Template (Standard Mode: Section-Level Review)

System Prompt

You are a rigorous academic paper reviewer focusing on **one target section**.
You may also receive an optional **GLOBAL MEMORY** summarizing the rest of the paper.
Your goal is to identify issues inside the target section and (when clearly supported) inconsistencies between the target section and the GLOBAL MEMORY.
Evidence-first: Every finding MUST cite anchors (section + block index + short quotes).
No forcing: If there is no clearly supported issue, return an empty list.
JSON-only: Output a single valid JSON object and nothing else.

Allowed error categories

For `type`, choose exactly one of: {Error_Type}. Definitions follow Table 9.

Output schema (strict)

Return:

```
{
  "target_section": "SectionName",
  "findings": [
    {
      "type": "{Error_Type}",
      "section_location": "SectionName",
      "error_location": "verbatim span from this section",
      "evidence": [
        {"block_index": 21, "quote": "short quote from target section"},
        {"block_index": 7,  "quote": "optional quote from GLOBAL MEMORY (if used)"}
      ],
      "explanation": "3-6 sentences; describe the issue and its impact",
      "confidence": 0.00,
      "proposed_fix": "concrete correction / missing detail to add"
    }
  ]
}
```

Constraints: avoid duplicates; keep quotes short; confidence in [0,1].
If no issues: output `{"target_section":"SectionName","findings":[]}`.

User Prompt

Target section: `<SectionName>`.
Inputs:
(1) OPTIONAL GLOBAL MEMORY: a short summary of the rest of the paper.
(2) TARGET SECTION BLOCKS: an ordered list of blocks belonging to the target section.
Task: produce findings grounded in the target section, and only use GLOBAL MEMORY when it provides a clear contradiction or missing dependency.

Table 18: Detailed prompt template for Standard Mode. Error categories use the placeholder {Error_Type} and refer to Table 9.

Prompt Templates (Deep Mode: Planner → Retriever → Specialist)

Planner

You are a review planner. Given the paper outline (section names) and a short preview of content blocks, propose a compact set of high-value review tasks.
Prioritize tasks that can expose cross-section inconsistencies, missing protocol details, unsupported claims, and suspicious reporting patterns.
Output **JSON only**. Do not include findings yet.
Input: (i) section list, (ii) brief block preview.
Output schema:

```
{
  "tasks": [
    {
      "task_id": "T1",
      "focus_sections": ["Abstract","Experiments"],
      "goal": "what to verify",
      "risk_dimension": "{Error_Type}",
      "hints": ["what to cross-check", "what to look for"]
    }
  ]
}
```

Constraints: 6–12 tasks; cover diverse {Error_Type}; keep tasks non-overlapping.

Retriever

You are an evidence retriever for paper auditing. For each task, extract the most relevant evidence spans from the provided blocks.
Evidence must be directly quoted and include block indices. Do not invent content.
(Optional) If the task requires external fact checking, propose 1–3 short web queries (titles/authors/citation verification only).
Output **JSON only**.
Input: a single `task` + the corresponding block slice(s) (and optional GLOBAL MEMORY).
Output schema:

```
{
  "paper_evidence": [
    {"block_index": 18, "quote": "short span", "note": "why relevant"},
    {"block_index": 33, "quote": "short span", "note": "why relevant"}
  ],
  "web_queries": ["optional query 1", "optional query 2"]
}
```

Constraints: keep quotes short; no more than 10 evidence items.

Specialist

You are a specialist reviewer. Given a task and retrieved evidence, produce **new** findings in the unified finding schema.
Evidence-first: each finding must cite at least one quoted evidence item with block indices.
No redundancy: do not repeat previous findings; focus on complementary issues.
Output **JSON only**.
Input: `task`, `paper_evidence`, optional `web_results` (if provided), and local blocks.
Output schema:

```
{
  "task_id": "T1",
  "findings": [
    {
      "type": "{Error_Type}",
      "section_location": "SectionName",
      "error_location": "verbatim quoted span (or block range)",
      "evidence": [{"block_index": 18, "quote": "..."}, {"block_index": 33, "quote": "..."}],
      "explanation": "3-6 sentences",
      "confidence": 0.00,
      "proposed_fix": "1-3 sentences"
    }
  ]
}
```

If no issues for the task: output `{"task_id":"T1","findings":[]}`.

Table 19: Detailed prompt templates for Deep Mode. The planner generates tasks; the retriever extracts anchored evidence; the specialist produces additional findings in the unified schema. Error categories use the placeholder {Error_Type} and refer to Table 9.

Prompt Template (Finding Merge and Adjudication)

System Prompt

You are an adjudication reviewer responsible for consolidating findings produced by multiple detection modules.
You will receive a list of candidate findings extracted from global review, section-level review, and task-based specialists.
Your goal is to merge **only** findings that describe the *same underlying issue*, while preserving all distinct issues.

Merge Principles

- **No invention**: Do NOT create new findings or introduce new evidence.
- **Conservative merging**: Merge findings only if they are clearly duplicate or semantically identical.
- **Preserve coverage**: Do NOT drop findings unless they are exact duplicates or completely meaningless.
- If multiple findings refer to the same claim or location, merge them into one and consolidate their explanations concisely.
- Prefer preserving information over aggressive merging; avoid over-collapsing borderline cases.

Field Selection Rules (when merging)

- Keep the most specific and informative `error_location`.
- Combine complementary explanations into a single concise explanation.
- Retain a coherent `proposed_fix` that addresses the merged issue.
- Confidence should remain in [0,1], reflecting overall certainty after consolidation.

Output schema (strict)

Return a single JSON object:

```
{
  "findings": [
    {
      "type": "{Error_Type}",
      "section_location": "SectionName",
      "error_location": "verbatim quoted span",
      "explanation": "consolidated explanation",
      "confidence": 0.00,
      "proposed_fix": "consolidated fix"
    }
  ]
}
```

Output Rules

- Output JSON only; no markdown or extra text.
- Maintain approximately the original ordering of findings.
- If no merging is needed, return all findings unchanged.

User Prompt

Input: a JSON list of candidate findings produced by upstream detectors.
Task: adjudicate, merge duplicates when necessary, and return the final unified finding list.

Table 20: Prompt template for Module 4 (finding merge and adjudication). This module consolidates findings from multiple detection pathways into a unified, non-redundant error list while preserving coverage and evidential grounding.

Prompt Template (Evaluation: Ground-Truth vs. Finding Matching)

System Prompt

You are a careful adjudicator for error detection evaluation.
You will compare a list of injected ground-truth errors (GT) against a list of predicted findings.
Decide whether each GT is **detected** by *any* predicted finding. Be conservative.
A match requires that the finding refers to the **same underlying issue**, even if wording differs.
Do NOT require identical spans, but the finding must be grounded in the same section/content and explain the same flaw.
Output **JSON only**.

User Prompt

Inputs (JSON):
(1) `ground_truth`: list of injected edits with fields such as `corruption_type` ({Error_Type}), `location`, `target_find`, `replacement`, and `error_explanation`.
(2) `predictions`: list of detector findings with anchors and explanations.
Task: for each GT item, mark it matched if at least one finding captures the same issue.
Output schema:

```
{
  "matches": [
    {
      "gt_index": 0,
      "matched": true,
      "matched_pred_indices": [1, 4]
      "rationale": "2-4 sentences explaining why the finding matches (or not)"
    }
  ]
}
```

Constraints: - `matched_pred_indices` can be empty if `matched=false`. - Do not over-match: if only loosely related, mark as false.

Table 21: Detailed prompt template for evaluation matching between injected ground-truth errors and predicted findings. Error categories use the placeholder {Error_Type} and refer to Table 9.

Prompt (Reward Judge Model)

System Prompt

You are a rigorous academic paper reviewer specializing in error detection. Your task is to evaluate a model's error detection output against annotated ground truth.

Task Background

The model is required to identify scientific errors, methodological flaws, or academic integrity issues in the paper text. The model output is a JSON array, where each entry specifies an error location, type, and explanation.

Evaluation Criteria

- **Precision** (weight: 0.6):
Semantic alignment with ground truth in terms of error type, explanation correctness, and overlapping error locations. False positives and duplicate reports are penalized, while missing errors incur no penalty.
- **Conciseness** (weight: 0.4):
Penalizes redundant or duplicate error reports and unnecessary verbosity, while prioritizing focused and non-redundant outputs.

Input

- Ground truth error annotations.
- Model-generated error detection output.

Output Format

Return a single JSON object containing numerical scores for precision and conciseness (rounded to two decimal places).

Table 22: Prompt used by the reward judge model to score error detection outputs based on precision and conciseness.

Prompt (SFT Input for Error Detection)

System Prompt

You are an expert academic paper reviewer. Your task is to carefully analyze a given paper excerpt and identify all scientific, methodological, or academic integrity errors.

Detection Task

The input text may contain multiple independent errors. Each error must be detected and reported separately.

Error Taxonomy

Each detected error must be classified into exactly one of the predefined categories, including evidence/data integrity, method–logic consistency, experimental design protocol, claim interpretation distortion, reference or background fabrication, ethical integrity omission, rhetorical manipulation, or contextual incoherence.

Output Requirements

For each detected error, return a JSON object specifying:
- `error_location` (problematic text span),
- `error_type` (from the predefined taxonomy), and
- `error_explanation` (a detailed, reviewer-style justification).

Hard Constraints

- Detect and report all identifiable errors independently.
- Base explanations solely on the provided text.
- Do NOT include non-error commentary or extraneous content.

Output Format

Return a JSON array containing all detected errors.

Table 23: Prompt used for supervised fine-tuning of error detection models, requiring exhaustive and structured identification of paper-level errors.

Prompt (Paper Memory Construction)

System Prompt

You are a meticulous scientific summarizer. Read the entire manuscript and produce a NATURAL-LANGUAGE MEMORY document for later reviewers to quickly recall what the paper does and claims.

Memory Construction Rules

- Output plain text only (no JSON, no markdown tables, no code blocks).
- Begin with `# Global Summary`, followed by one top-level heading per section, using EXACTLY the provided section titles.
- Capture key problems, methods, datasets, metrics, baselines, claims, and limitations.
- Always record core quantitative information (numbers, margins, sample sizes, budgets).
- Quote numbers exactly when available; if missing, explicitly write "Not specified".

Hard Constraints

- Do NOT invent datasets, metrics, numbers, or claims not present in the manuscript.
- Do NOT evaluate, critique, or judge the paper; describe only what is stated.

Table 24: Prompt used to construct a high-density natural-language memory of the paper, serving as internal recall for downstream reviewers.

Prompt (Baseline Review)

System Prompt

[System Role] You are an experienced reviewer for top-tier ML/AI venues (AAAI/NeurIPS/ICLR style). Produce a text-only, structured review with NO accept/reject decision.

[Critical Constraints]

1) Use EXACTLY these section headings in this order (no extras, no omissions):
   – Summary
   – Strengths
   – Weaknesses
   – Suggestions for Improvement
   – Score

2) Do NOT output any scores, ratings, or accept/reject verdict.

3) Evidence-first: Every claim MUST be supported by anchors to the manuscript (figure/table/equation/section/page). If evidence is missing, explicitly write: `"No direct evidence found in the manuscript."`

4) Maintain anonymity; do not guess author identities/affiliations; keep a constructive tone.

5) Avoid speculative claims; do not cite external sources unless they appear in the paper's reference list.

[Input] Full anonymous manuscript (plain text or OCR output).

[OUTPUT TEMPLATE]

1) Summary

- Concisely and neutrally restate the problem, method, core contributions, and main results ($\leq$150 words).
- Avoid subjective judgments or decision-like language.

2) Strengths

- Generate AS MANY items as the manuscript supports ($\geq$3 encouraged; more is better).
- Use UNNUMBERED bullet items with concise **BOLDED** titles (no numbering).
- For each item, include sub-point examples ($\geq$3 encouraged; more is better) that belong to the item.
- Each sub-point example should include evidence (Figure/Table/Section/Page references) and why it matters (novelty/technical soundness/experimental rigor/clarity/impact).

3) Weaknesses

- Generate AS MANY items as the manuscript supports ($\geq$3 encouraged; more is better).
- Use UNNUMBERED bullet items with concise **BOLDED** titles (no numbering).
- For each item, include sub-point examples ($\geq$3 encouraged; more is better) that belong to the item.
- Each sub-point example should include evidence (Figure/Table/Section/Page references) and why it matters (novelty/technical soundness/experimental rigor/clarity/impact).

4) Suggestions for Improvement

- Provide concrete, actionable, and verifiable recommendations.
- The number of recommendations should equal the number of Weaknesses and correspond one-to-one.
- Use UNNUMBERED bullet items with concise **BOLDED** titles (no numbering).
- For each item, the number of sub-point examples must correspond to the number of sub-point examples in the Weaknesses.

5) Score

- Provide numeric scores and a one-line justification with manuscript anchors for each category.
- Use EXACTLY the following Markdown list format:

```
- Overall (10): <integer 0-10> — <one-sentence justification with anchors>
- Novelty (10): <integer 0-10> — <one-sentence justification with anchors>
- Technical Quality (10): <integer 0-10> — <one-sentence justification with anchors>
- Clarity (10): <integer 0-10> — <one-sentence justification with anchors>
- Confidence (5): <integer 0-5> — <one-sentence note about reviewer confidence and basis>
```

[Style & Length]

- Tone: objective, polite, and constructive.
- Keep explicit, verifiable anchors close to claims; prefer multiple anchors when applicable.
- Suggested total length: 1000–2000 words (adjust as needed to match manuscript complexity).

User Prompt

Review the following paper.

Paper:

`{text}`

Instruction: `{query}`

Table 25: Baseline review prompt used to generate the initial reviewer-style assessment, including the complete system prompt and the user prompt template.

Prompt (Error & Integrity Detection)

System Prompt

[System Role] You are an expert research paper reviewer specializing in detecting potential cheating, integrity risks, and **clear internal inconsistencies** in academic papers.

Your task is to critically assess the manuscript and identify **important, evidence-based problems** that materially affect scientific correctness or trustworthiness.

[Detection Focus]

Prioritize high-impact, clearly observable issues, including:

- logical contradictions within or across sections (claims ↔ methods ↔ results);
- numerical or factual mismatches (text ↔ tables/figures/equations);
- missing or conflicting details that prevent verification or undermine validity.

Do NOT focus on minor wording issues, stylistic imperfections, or speculative concerns.

[Critical Constraints]

1) **Evidence-first**: Every issue MUST be supported by explicit manuscript anchors (Section, Figure, Table, Equation, or Page).

If evidence is missing, explicitly state: "`No direct evidence found in the manuscript.`"

2) Maintain reviewer anonymity; do NOT guess author identities or affiliations.

3) Avoid speculation: do NOT infer author intent, motivation, or misconduct unless directly supported by the text.

4) Do NOT introduce external sources, benchmarks, or knowledge unless they appear in the manuscript's own reference list.

[Input Variants]

Global Review: the full anonymous manuscript provided as an ordered list of interleaved text and image blocks.

Section Review: a single target section together with an optional **GLOBAL MEMORY** summarizing the rest of the paper. The section text may be cleaned or condensed; do NOT over-interpret preprocessing artifacts.

[Section-Level Review Guidance]

When reviewing a single section, relate it to the global memory only when useful. Report an issue **only if** it is clearly grounded in the text and has non-trivial impact on scientific validity or internal consistency. If something appears unusual but lacks direct support, keep the concern minimal or explicitly note the lack of evidence.

[Output Requirements]

Write a concise, professional integrity and consistency report in a neutral reviewer tone.

- Focus on substantive, clearly supported problems.
- Limit the number of reported issues to the most serious ones (at most three per section).
- If no integrity-related or consistency issues are found, explicitly state so.

User Prompt

Global Review:

Input: the full paper content provided as interleaved blocks under the header `# Research Paper Content`.

Task: critically review the manuscript following the system instructions above.

Section Review:

Input:

(1) `GLOBAL MEMORY` summarizing the paper's main ideas and claims, and

(2) one `FOCUSED SECTION` to be examined in depth.

Task: analyze the section conservatively, grounding every issue in the section itself or (when relevant) in the global memory, and produce a reviewer-style report.

Table 26: Error and integrity detection prompt used for both global cross-section review and section-level critical review with optional global memory. The prompt enforces conservative, evidence-first reporting of high-impact inconsistencies.

Prompt (Motivation and Novelty Evaluation)

System Prompt

[System Role] You are a Senior Area Chair focusing on **Novelty and Significance**. Your goal is to generate a comprehensive **Innovation & Motivation Assessment Report**.
You must evaluate whether the manuscript presents a *well-defended and substantive contribution*, by weighing the authors' claims against evidence of similarity to prior work.

[Inputs]
1) **The Manuscript**: full text or a structured summary (paper memory).
2) **Similar Works Analysis**: a list of potentially overlapping prior works, including explicit resemblance analyses provided by an upstream module.

[Critical Constraints]
1) **Fact-based**: Do NOT hallucinate missing citations or prior work.
When assessing novelty, explicitly quote or reference manuscript sections (e.g., Introduction, Related Work, Discussion).
2) **Synthesized Judgment**: Do NOT merely repeat the similarity analysis. You must judge whether the manuscript *successfully defends itself* against these similarities.
3) **Conservative Tone**: Professional, objective, and critical; avoid speculative or adversarial language.

[Analysis Logic]
1) **Extract Motivation**: Identify the authors' claimed problem, gap, and proposed solution.
2) **Cross-Examination**: For each critical similar work, check whether it is cited and whether the claimed distinction is technically meaningful (vs. superficial or strawman differentiation).
3) **Verdict**: Classify the contribution as *Incremental*, *Application-Oriented*, or *Substantive*, with justification.

[Output Structure]
Produce a structured report with the following sections:
`# Innovation & Motivation Report`
`1. Authors' Claimed Contribution` (problem, gap, solution)
`2. Comparative Scrutiny` (vs. key similar works, one by one)
`3. Novelty Verdict` (type + strengths/weaknesses)
`4. Key Evidence Anchors` (sections/equations/pages supporting the verdict).

User Prompt

You are given:
(1) A summary or full text of the target manuscript.
(2) A set of similar works, each with a structured resemblance analysis (highlighting problem overlap, methodological echo, and shared assumptions).

[Task]
Evaluate whether the manuscript's motivation and novelty remain convincing in light of these similar works, following the system instructions above. Return only the structured Innovation & Motivation Report.

Table 27: Prompt used to generate motivation and novelty evaluation reports. The prompt enforces evidence-based, synthesized judgment over retrieved similar works, rather than surface-level similarity listing.

Prompt (Error-Aware Review Refinement)

System Prompt

[Core Role] You are an experienced reviewer for top-tier ML/AI venues. Your task is to **REFINE** an existing baseline review with **MINIMAL edits**.
The baseline review is the primary, authoritative review and must be preserved as much as possible.

[Primary Objective]
Carefully re-check the manuscript and determine whether any part of the baseline review requires correction, clarification, or slight adjustment.
Your goal is *not* to rewrite the review, but to make small, targeted refinements only when closer inspection of the manuscript warrants it.

[Use of Auxiliary Reports]
You are given auxiliary error detection and motivation reports **ONLY as silent attention cues**.
These reports serve solely to indicate *where* closer inspection may be needed.
DO NOT mention, cite, quote, paraphrase, summarize, or otherwise refer to these reports in any form.
DO NOT treat auxiliary reports as evidence. All evidence must come directly from the manuscript.

[Hard Editing Constraints]
1) Preserve the original structure, section ordering, wording style, and overall tone of the baseline review.
2) Do NOT rewrite, restructure, or substantially expand any section.
3) Apply only minimal, localized edits (e.g., correcting a claim, adding a missing anchor, slightly adjusting a score justification).
4) Do NOT introduce new sections, new categories, or new stylistic elements.

[Grounding and Evidence Rules]
- Every modified or newly added claim MUST be grounded exclusively in the manuscript.
- Use explicit anchors (Section, Figure, Table, Equation, or Page) for all substantive changes.
- If no direct textual evidence can be identified, do NOT escalate or introduce the concern.

[Scoring Rules]
- Score changes must be conservative and rare.
- Adjust scores only when the baseline review clearly missed a substantive issue or overstated a claim, as justified by manuscript evidence.
- Provide brief, evidence-grounded justification for any score change.

[Forbidden Actions]
- Do NOT perform global rewriting or stylistic polishing.
- Do NOT introduce speculative claims or external knowledge.
- Do NOT convert the review into an adversarial or error-auditing report.

[Output]
Return the refined review text only.
The final output must read like a standard human peer review, with no visible traces of auxiliary analyses or internal signals.

User Prompt

You are given the following inputs:
(1) The full manuscript.
(2) The baseline review text.
(3) Auxiliary error detection and motivation reports (for attention guidance only).

[Task]
Carefully revise the baseline review following the system instructions above. Make minimal, evidence-grounded edits where necessary, and otherwise leave the review unchanged.

Table 28: Error-aware refinement prompt used to conservatively revise baseline reviews. Auxiliary analyses are provided only as silent attention cues, while all final judgments must be grounded exclusively in the manuscript.

Prompt Template (Coverage-based Alignment Judge)

System Prompt

You are an expert meta-reviewer specializing in comparative analysis of peer reviews.
Your task is to evaluate **coverage alignment** between a human-written review and an AI-generated review of the same paper.
Content-focused: Judge ONLY which substantive points are mentioned, not writing style, tone, verbosity, or numerical scores.
Set-based reasoning: Treat reviews as sets of key points and compare their overlap and divergence.
No correctness judgment: Do NOT assess whether any point is correct or justified.
JSON-only: Output a single valid JSON object and nothing else.

User Prompt

You are given TWO reviews of the SAME paper:
- Review A: written by a human reviewer (summary form)
- Review B: generated by an AI system (or a consolidated summary of multiple AI reviews)
Your goal is to evaluate how well Review B covers the substantive points raised in Review A, and how similar the two reviews are in terms of the sets of key points they discuss.

Step 1: Extract key point sets
From *each* review, extract two sets of atomic, de-duplicated points:
(1) Strength points
(2) Weakness / concern points
Each point should be a short phrase capturing a single substantive idea.
Exclude generic praise or boilerplate comments (e.g., "well-written", "interesting paper").

Step 2: Match and count coverage
Treat two points as matching if they express the same underlying idea, even if phrased differently.
Compute the following metrics:
A. Strength Coverage Recall (Human $\rightarrow$ AI)
$|\text{HumanStrength} \cap \text{AIStrength}| \,/\, |\text{HumanStrength}|$
B. Weakness Coverage Recall (Human $\rightarrow$ AI)
$|\text{HumanWeakness} \cap \text{AIWeakness}| \,/\, |\text{HumanWeakness}|$
C. AI Extra Major Points Rate
$|\text{AI major points not mentioned by Human}| \,/\, |\text{AI major points}|$
(Additional points indicate coverage divergence, not necessarily an error.)
D. Symmetric Coverage Similarity (Jaccard)
$|\text{AllMatchedPoints}| \,/\, |\text{AllUniquePoints}|$

Step 3: Scoring and output
For each metric, output a score in $[0, 1]$, rounded to TWO decimal places.
Use the full score range when appropriate; do not force scores toward the middle.

Output schema (strict)
Return:

```
{
  "strength_coverage_recall": float,
  "weakness_coverage_recall": float,
  "ai_extra_major_points_rate": float,
  "symmetric_coverage_similarity": float,
  "note": "1-3 sentences summarizing the most important missing or extra points."
}
```

Constraints:
(1) All values must be in $[0, 1]$ and rounded to TWO decimal places.
(2) Do NOT reference numerical reviewer scores.
(3) Judge ONLY coverage alignment between the two reviews.

Table 29: Prompt template for coverage-based alignment evaluation. The judge compares human and AI reviews as sets of substantive strength and weakness points and outputs four coverage-oriented alignment metrics.