Analysis of Results in the ML Research Field

How well can an LLM decide the reproducibility of a paper?

Andrei Opritoiu[1]

Supervisor(s): David M.J Tax[1]**, Chenxu Hao**[1]**, Hayley Hung**[1]**, Nergis Tömen**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Andrei Opritoiu
Final project course: CSE3000 Research Project
Thesis committee: David M.J Tax, Chenxu Hao, Hayley Hung, Nergis Tömen, Klaus Hildebrandt

An electronic version of this thesis is available at https://gitlab.tudelft.nl/cse3000/analysisMLresults.

Abstract

The recent surge in machine learning (ML) research has led to a record number of paper submissions, overwhelming the traditional peer-review process. Although conferences like NeurIPS have introduced reproducibility checklists to maintain scientific standards, manual verification of these claims is time-consuming and inconsistent. This study investigates the feasibility of using Large Language Models (LLMs) to automate the evaluation of paper reproducibility. By creating a ground-truth dataset through the manual annotation of NeurIPS papers, this study assesses the accuracy of LLMs in verifying author claims regarding code availability, hyperparameter transparency, and compute resources. The results compare LLM performance with manual labels to identify where automated tools succeed and where they fail to capture technical nuances. Ultimately, this research demonstrates that while LLMs can act as highly efficient administrative filters to streamline initial screening, they fail to reliably predict execution viability, highlighting the remaining boundaries of automated verification.

1 Introduction

As Large Language Models (LLMs) demonstrate increasing capabilities in reasoning and document analysis, there is growing interest in using these tools to automate bottlenecked parts of the scientific peer-review process. An area of particular concern in the machine learning community is reproducibility [4]. This paper explores whether the very models responsible for the current AI research boom can be deployed to audit the transparency and reliability of the literature they helped create.

The underlying motivation for this research comes from a structural crisis in the machine learning community. The velocity of ML development has triggered an exponential growth in paper submissions. This surge has overwhelmed the academic community's capacity for thorough peer review, frequently resulting in the publication of papers lacking crucial execution data. While top-tier conferences have attempted to self-regulate by mandating "Reproducibility Checklists," human reviewers rarely have the time to download repositories, check hyperparameter bounds, or verify hardware constraints. Automating this auditing process via LLMs is a highly promising solution due to their advanced text-mining and semantic retrieval capabilities.

However, a critical systemic risk remains unaddressed: automation bias. If an LLM is tasked with auditing a paper, it may simply take the path of least resistance by parsing and trusting the author's self-reported checklist statements rather than cross-examining the main body text to confirm those claims are true. If an AI auditor blindly agrees with an author's fabricated justifications, it defeats the entire purpose of an independent audit, ultimately degrading scientific integrity rather than protecting it. Understanding exactly where and why an LLM fails or succeeds as an auditor is vital. If LLMs can be proven reliable, they can act as an automated first-line defense filter for reviewers that instantly flag papers that lack functional source links or compute documentation before a human reviewer even opens the file.

This thesis seeks to evaluate the reliability of LLMs as reproducibility auditors by addressing the following research sub-questions:

- Does the LLM falsely label papers as "Reproducible" simply because the authors' self-claims say so, rather than verifying the evidence?
- How accurately can the LLM locate and verify code URLs, dataset citations, hyperparameter tables, and hardware descriptions within a paper?
- Can the LLM distinguish vague methodology descriptions and assess the true reproducibility of a paper?

The remainder of this paper is structured as follows. Section 2 reviews the related work concerning peer review automation and the reproducibility crisis. Section 3 defines the formal extraction task and outlines our research methodology. Section 4 details the design of our LLM evaluation pipeline. Section 5 presents our experimental setup and reports the qualitative results for each sub-question. Section 6 addresses responsible research practices and ethical considerations. Finally, Section 7 provides a discussion of our findings in a broader context, and Section 8 concludes the paper with directions for future work.

2 Related Work

The existing state of the art thoroughly investigates the ongoing reproducibility crisis within machine learning, noting persistent missing components such as missing code repositories, raw data, and hyperparameter specifications [4; 8; 7]. To alleviate peer-review bottlenecks, initial studies on "ReviewerGPT" frameworks demonstrate that LLMs can provide helpful high-level review text and feedback [5]. Consequently, dedicated automated auditing frameworks, such as AI Reproducibility Copilots [1] and specialized "Commitment Checklists" [2], have emerged to automatically parse manuscripts and verify whether the authors actually deliver on their technical execution promises.

However, existing trials reveal critical vulnerabilities in automated review pipelines. For example, during the NeurIPS 2024 automated checklist experiment, researchers observed that authors could successfully "game" LLM checklist assistants by utilizing fabricated justifications that the model failed to verify or cross-examine properly [3]. Furthermore, the accuracy of these automated auditing tools is highly sensitive to the academic domain, performing unevenly across different fields like computer science versus specialized medical or transplantation research [9; 2].

While the literature establishes that LLMs can find high-level metadata but are easily misled by author claims [3], several critical vulnerabilities remain unexplored. First, regarding the blind trust paradox (Research Question 1, or RQ1), it remains unanswered how systematically LLMs succumb to

automation bias by relying on explicit author declarations instead of independent textual extraction. Prior work has not quantified the model's shortcut bias when facing pre-filled checklists; this study tests whether instructing the LLM to distrust these claims forces text-grounded validation. Second, concerning granular extraction (RQ2), existing studies focus on high-level metadata, leaving a gap in understanding whether LLM detection accuracy remains uniform when searching for structured parameter tables versus unstructured data strings like hardware compute configurations or active repository links. Third, regarding holistic evaluation (RQ3), current frameworks primarily focus on the superficial extraction of elements [2], but it remains unknown whether an LLM can synthesize these extracted artifacts to accurately assess the overall reproducibility of a manuscript relying strictly on its core textual content.

3 Problem Definition & Dataset

This section outlines the formal parameters of the automated auditing process and details the distinct datasets curated to evaluate the model's performance against human benchmarks.

3.1 Task Definition

We formally define the automated reproducibility extraction task as a targeted information retrieval and verification problem. Given a scientific manuscript, the objective is to deploy an LLM to accurately identify, extract, and validate specific reproducibility markers embedded within the text. For the scope of this research, we restrict these markers to three primary categories: Open Access (e.g., active repository links, dataset availability), Compute Resources (e.g., hardware constraints, processing unit specifications), and Experimental Settings (e.g., hyperparameter configurations, seed values).

3.2 Dataset Collection and Annotation

To evaluate the LLM across different dimensions of the reproducibility extraction task, we curated three distinct evaluation datasets. Each dataset was manually annotated by human reviewers to establish a definitive "ground truth" benchmark against which the automated pipeline's outputs were measured.

- **Adversarial Checklist Dataset (RQ1):** To test the model's susceptibility to automation bias and blind trust, we designed an adversarial dataset consisting of 30 test cases evenly split across our three categories (10 papers per category). We intentionally introduced specific contradictions between the author-provided reproducibility checklists and the actual manuscript text:
  - *Open Access:* Introducing 10 cases where the checklist claimed a valid repository link exists, but the link was non-existent.
  - *Compute Resources:* Introducing 10 cases where the checklist falsely claimed compliance ("Yes") when the actual text lacked these details ("No").
  - *Experimental Settings:* Utilizing 10 baseline control cases where the checklist claimed compliance ("Yes") and the underlying text accurately supported it ("Yes"), verifying if the model can accurately validate true positives without defaulting to distrust.
- **Granular Extraction Dataset (RQ2):** To assess the LLM's accuracy in locating and verifying specific artifacts, we expanded our core dataset to include 76 accepted papers from the NeurIPS Main Conference Track. These manuscripts were manually reviewed to determine their actual compliance regarding Open Access, Compute Resources, and Experimental Settings. Furthermore, to establish the human-annotated ground truth for the holistic reproducibility score evaluated in our extended Q2 experiment, we assessed these manuscripts against a strict tripartite rubric:
  - *Strong Reproducibility:* The manuscript provides comprehensive access to all necessary components, including functional open-access repository links, explicit compute resource constraints, and exact experimental settings (e.g., hyperparameter tables and seed values) required to recreate baseline results.
  - *Partial Reproducibility:* The manuscript provides the majority of necessary components but suffers from minor omissions (e.g., code and settings are available, but specific hardware constraints are missing, requiring the reproducer to guess the optimal environment).
  - *Weak Reproducibility:* The manuscript is missing multiple critical execution data, lacks functional repository links, fails to disclose vital hyperparameters, or relies on vague methodological descriptions, rendering independent reproduction effectively impossible.
- **Reproducibility Benchmark Dataset (RQ3):** To investigate whether the LLM can accurately assess overall reproducibility against empirical, real-world execution, we utilized a subset of papers previously evaluated by the Machine Learning Reproducibility Challenge (MLRC). Because these papers have undergone rigorous, independent manual testing to confirm whether their findings can actually be reproduced, they provide a highly reliable baseline to compare against the LLM's automated grading.

4 Approach & Design

This section details the architecture of our automated evaluation system and provides the rationale behind our technical design choices, including model selection, prompt engineering, and document parsing strategies.

4.1 Pipeline Components & Strategy

The LLM-based Reproducibility Evaluation Pipeline is designed to process scientific manuscripts and output structured reproducibility assessments. The pipeline consists of three primary components:

- Document Pre-processing: We first process the raw portable document format (PDF) of the accepted

NeurIPS paper through a dedicated trimming module. This trimmer removes the author-provided reproducibility checklist from the document, outputting a new, checklist-free PDF.

- Extraction Engine: The selected PDF is fed directly into the LLM alongside a structured prompt. The engine evaluates the document against the three predefined criteria: Open Access, Compute Resources, and Experimental Settings.
- Output Generator: For each criterion, the pipeline requires the model to output a standardized response format containing a binary reproducibility verdict, a detailed justification, and the exact textual quote extracted from the paper to serve as verifiable evidence.

4.2 Design Choices & Justifications

- **Model Selection:** We utilized the `gemini-3.0-flash-preview` model via the Google AI Studio API. This model was selected for its exceptionally large context window—enabling single-pass processing of entire manuscripts and its advanced semantic and multimodal reasoning capabilities within a cost-effective, free-tier framework.
- **Data Ingestion Format:** Initial attempts to convert source PDFs to Markdown (`.md`) to optimize inference time revealed a critical flaw: the conversion stripped graphical elements like charts and figures. Because the Gemini model is inherently multimodal, this loss of visual layout data degraded extraction accuracy, particularly for experimental parameters embedded within graphs. Given that empirical testing showed no discernible processing time advantage for Markdown text over raw PDFs, we finalized our design to ingest original PDFs directly, fully preserving visual data integrity.
- **Prompt Design:** Rather than a single generalized instruction set, we engineered a modular prompting architecture using three distinct prompts (`LLM_PROMPT_OPEN_ACCESS`, `LLM_RESOURCE_COMPUTE`, and `LLM_SETTINGS`) to isolate each reproducibility metric. This design leverages advanced prompt engineering to maximize grounding and mitigate automation bias:
  - **Role-Prompting and Persona Assignment:** By instructing the LLM to act as an "expert peer-reviewer for the NeurIPS conference," the prompt forces the model to evaluate the manuscripts using strict academic standards and machine learning evaluation rigor.
  - **Explicit Negative Constraints:** To counter the "blind trust paradox" and automation bias, we embedded a prominent negative constraint within the prompt core: *'DO NOT USE THE AUTHORS ANSWER AS A VALID ANSWER, YOU MUST CHECK FOR YOURSELF!'*. This explicitly prevents the model from treating author checklist self-claims as ground truth, forcing independent document cross-examination.
  - **Domain-Specific In-Context Rubrics:** Avoids open-ended evaluation by injecting official, literal NeurIPS checklist questions and formatting guidelines (e.g., restricting outcomes strictly to categorical classification of *yes*, *no*, or *na*), bounding judgment to exact academic compliance standards.
  - **Deterministic Output Structuring (JSON Integration):** Facilitates automated downstream parsing and eliminates conversational text by forcing a strict JSON template response. The schema enforces precise variables: `paper_title` (strictly lowercase with underscores), the targeted checklist verification field, an `exact_quote` parameter, and a `justification` parameter.
  - **Verbatim Grounding Mechanisms:** To combat semantic hallucination, the JSON schema introduces a verification constraint requiring an `exact_quote` copy-pasted directly from the manuscript to support any *yes* classification, or `NONE` if no text exists, forcing text-grounded self-verification before finalizing judgments.

5 Results

This section presents the quantitative and qualitative outcomes of the LLM evaluation, assessing its susceptibility to automation bias, its granular extraction accuracy, and its holistic reproducibility grading.

5.1 Experimental Setup

The automated evaluation pipeline was developed in Python and executed locally on macOS utilizing an Apple Silicon M4 architecture. Inference was conducted by querying the `gemini-3.0-flash-preview` endpoint via the Google AI Studio API.

5.2 RQ1: Automation Bias and Blind Trust

Evaluation of the 30-paper adversarial dataset revealed a high susceptibility to automation bias when conflicts existed between author claims and actual manuscript text.

- **Explicit Contradictions (Open Access):** When confronted with false repository URLs, the LLM extracted evidence directly from the fabricated checklist, ignoring the main text in 80% (8 out of 10) of instances (see Figure 4a).
- **Nuanced Omissions (Compute Resources):** When authors omitted required hardware details, the LLM split its reliance evenly (50/50) between the flawed checklist and the main text (see Figure 4c).
- **Truthful Alignment (Experimental Settings):** When author claims truthfully aligned with the text, the model correctly bypassed the checklist to extract quotes from the methodology sections in 100% of instances (see Figure 4e).

Applying explicit prompt warnings (e.g., "DO NOT USE THE AUTHORS ANSWER AS A VALID ANSWER, YOU MUST CHECK FOR YOURSELF!") failed to physically

prevent the model from navigating to the checklist (see Figure B). However, the constraint increased the critical nature of the generated justifications, allowing the model to synthesize the correct overall binary verdict in 80% (24 out of 30) of cases. Ultimately, structural trimming (physically removing the checklist prior to inference) was the only intervention that successfully forced exclusive text-grounded validation. Furthermore, our analysis revealed that the quality and granularity of the prompt guidelines significantly impact evaluation accuracy; highly detailed rubrics yielded notably better performance.

We also observed that the LLM's capability varies depending on the task type. While it excels at extracting explicit textual data, such as experimental settings, it struggles with metrics like Open Access. This discrepancy stems from a fundamental technical limitation: the LLM cannot actively browse external links or open supplemental materials to independently verify code and repository availability, forcing it to either rely on author claims or fail the verification. Therefore, maximizing the model's accuracy requires explicit instructions detailing exactly what artifacts must be extracted, alongside clear protocols for handling ambiguous or confusing text.

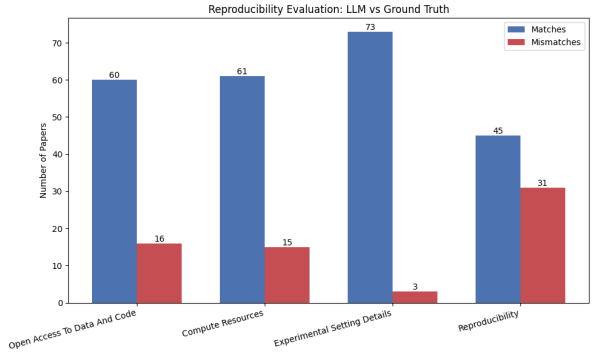
5.3 RQ2: Granular Extraction Accuracy and Holistic Synthesis

To assess the model's capacity for granular verification at scale, we analyzed its extraction performance across a final dataset of 76 successfully parsed NeurIPS papers. Due to API rate limits, free-tier token constraints, and processing time limitations, our extraction methodology for this expanded dataset required adaptation. Rather than querying the model with three isolated, highly specific prompts, we merged the Open Access, Compute Resources, and Experimental Settings criteria into a single, comprehensive prompt.

Overall, the LLM successfully located and verified Open Access elements (e.g., active repository links and dataset citations) with an accuracy of 79% (60/76), while Experimental Settings (e.g., hyperparameter tables and seed values) were correctly extracted in an impressive 96% (73/76) of cases. Performance on Compute Resources yielded 80% accuracy (61/76). See Figure 1a.

Crucially, while the data indicates that the LLM performs exceptionally well at identifying highly structured textual variables (like settings), we observed an unexpected degradation in accuracy specifically tied to our prompt merging. For metrics like Compute Resources, the confusion matrix reveals a notable bias toward False Negatives, the model frequently predicted "no" when the ground truth was "yes." (see Appendix: C) We theorize that this performance drop is a direct result of the expanded prompt length and the complexity of multi-tasking. Forcing the LLM to process a massive set of instructions while simultaneously searching for three distinct categories of evidence appears to dilute its attention, causing it to skip over hardware details that it successfully extracted when prompted in isolation.

Furthermore, we extended this specific experiment to test whether the LLM could synthesize these three extracted metrics into a final, holistic reproducibility score (categorized in our human-annotated ground truth as *Strong*, *Partial*, or *Weak*). When tasked with evaluating this overall reproducibility factor based on the extracted artifacts, the LLM's accuracy dropped to 59% (45 matches and 31 mismatches). This steep decline demonstrates that while the model is highly capable of retrieving isolated data points, the complex semantic reasoning required to weigh multiple imperfect variables and assign a conclusive academic grade remains a significant vulnerability. Crucially, this 59% accuracy baseline represents an evaluation limited strictly to static textual claims; it highlights a fundamental gap between a model's ability to verify the *presence* of a reproducibility artifact versus its ability to judge true reproducibility. In practice, a manuscript can flawlessly document its parameters and link to a public repository on paper, yet remain entirely unreproducible due to silent software errors, undocumented dependencies, or broken source code that only manifests during execution. An LLM reading a static document is completely blind to these functional realities, meaning its evaluation measures the *promise* of reproducibility rather than its technical reality. Therefore, to determine the model's true capability of assessing reproducibility we pivot to human-verified papers where independent researchers have physically run the code, a challenge we address directly in our final research question.



(a) Overall Extraction Accuracy

Figure 1: Granular extraction performance across 76 NeurIPS papers using merged prompting.

5.4 RQ3: Overall Reproducibility Assessment Against Empirical Benchmarks

Given the limitations of text-only synthesis, we deployed our evaluation pipeline against a curated benchmark of papers from the Machine Learning Reproducibility Challenge (MLRC [6]). Unlike the NeurIPS dataset, where the ground truth was inherently limited to the presence of written claims, the MLRC dataset provides an uncompromising baseline: independent researchers physically downloading the code, setting up environments, and attempting to replicate the empirical findings. This allowed us to explicitly test whether the LLM's automated grading correlates with functional, real-world execution.

When tasked with predicting overall reproducibility outcomes against this empirical baseline, the LLM matched

the human empirical outcomes in only 35% (14 out of 40) of cases, as detailed in the Predicted Reproduction Outcome matrix (Figure 3c). While the LLM easily reads and validates the *architectural description* of these frameworks within the manuscript, it remains fundamentally blind to runtime anomalies. The confusion matrices and specific metrics profiles in Figure 3 confirm a severe, systemic trend: the model suffers from extreme optimization optimism, dramatically underestimating technical execution risks; this can easily be seen in Figure 3a and Figure 3b, by looking at how many papers (26 in Figure 3a and 20 in Figure 3b) the LLM estimated a much lower risk compared to the human ground truth.

Furthermore, this tracking blindness is most acute in Figure 3a (*Code and Environment Risk*), where the model achieved an execution-matching accuracy of only 17.5% (7 matches vs. 33 mismatches). The LLM classified the vast majority of environments as "low_risk," completely oblivious to the fact that human replicators encountered fatal environmental constraints or severe code issues during physical deployment. Similarly, as shown in Figure 3c, while human trials yielded 23 instances of hard-fought "partial_success," the LLM blindly assigned an outright "success" verdict to 21 of those cases.

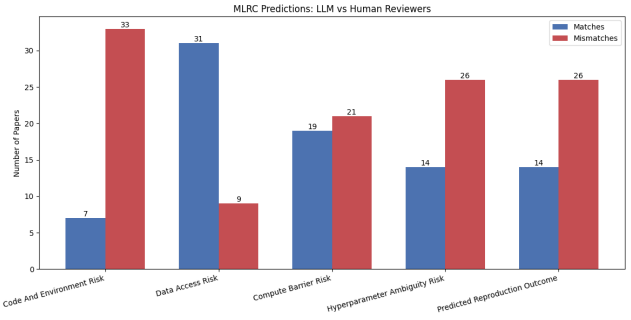
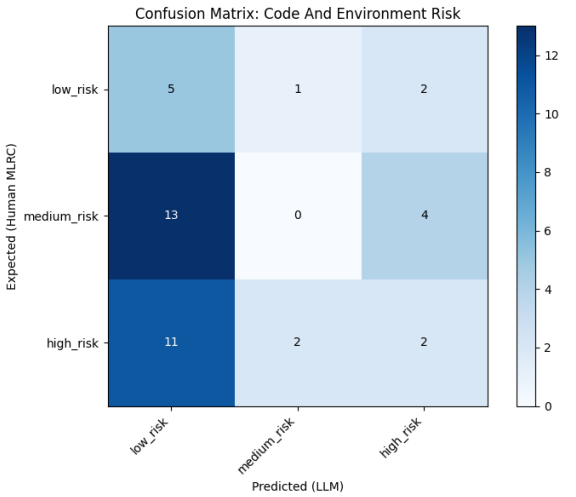
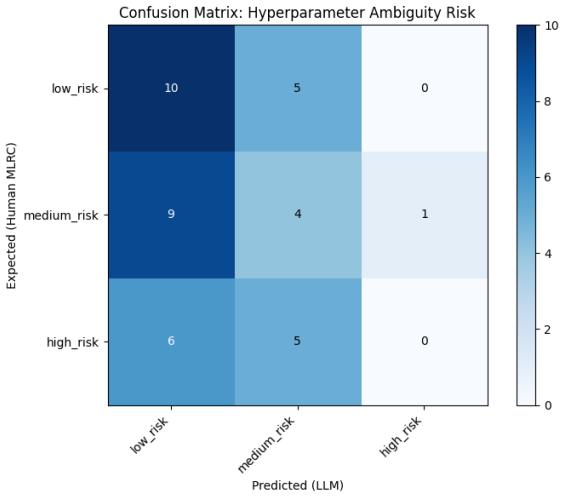


Figure 2: Overall evaluation metrics tracking exact LLM prediction matches vs. human mismatches across all evaluated risk categories on the MLRC dataset.

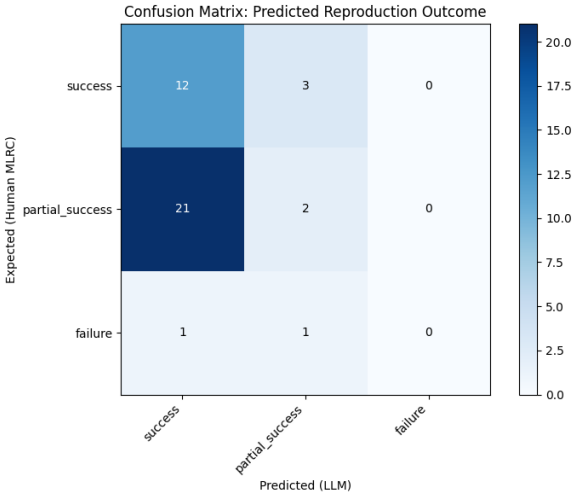
Ultimately, these findings expose the boundaries of using large language models as autonomous peer-review auditors, aligning directly with the core procedural barriers identified by Semmelrock et al. [8] As they contend, the general machine learning community often underestimates how fragile end-to-end outcome reproducibility can be due to poor adherence to documentation standards and the hyper-sensitivity of ML training conditions. Our pipeline's performance proves that while an LLM operates with near-perfect accuracy when verifying structural checklists or extracting rigid hyperparameter matrices, it exhibits a severe blind spot regarding functional viability. The model evaluates the architectural *promise* of compliance written within the manuscript text; it cannot predict the silent infrastructure failures, code rot, or package dependency conflicts that only surface when code is being executed (summarized in Figure 2). Consequently, automated LLM auditing serves as an exceptional pre-screening filter to catch completely undocumented or unlinked work, but human



(a) Code and Environment Risk



(b) Hyperparameter Ambiguity Risk



(c) Predicted Reproduction Outcome

Figure 3: MLRC Confusion Matrices

execution remains irreplaceable for verifying genuine scientific reproducibility.

6 Responsible Research

Pursuing automated academic peer-reviewing tools introduces unique ethical obligations, resource considerations, and data responsibilities. This section reflects on the ethical implications of deploying large language models (LLMs) within scientific evaluation frameworks and provides a rigorous disclosure of our own methodology's reproducibility.

6.1 Ethical Considerations and Potential Risks

The primary ethical concern identified in this research is the *automation bias paradox* documented in our RQ1 and RQ2 experiments. If academic publishers or conference chairs deploy LLM-based verification architectures as autonomous gatekeepers without human oversight, the system risks creating an unfair, superficial compliance culture. Because the pipeline evaluates the structural *promise* of a manuscript rather than its functional integrity, it inadvertently creates an environment where authors can optimize text descriptions to bypass automated screeners while hosting broken, unexecutable code.

Furthermore, relying heavily on commercial, proprietary API models introduces data privacy risks and proprietary vulnerabilities. Inadvertently submitting unreleased, under-review manuscripts to external model logging loops poses a severe intellectual property risk for researchers. Finally, the carbon footprint and computational costs associated with passing entire multi-page scientific manuscripts through large context window models must be balanced against the perceived efficiency gains of automated grading.

6.2 Methodological Reproducibility

To ensure that our own work adheres to the highest standards of the reproducibility guidelines explored in this paper, we disclose our complete operational environment, artifact links, and execution rubrics:

- **Hardware and Computing Infrastructure:** All parsing and inference workflows were executed locally on an Apple MacBook Pro equipped with an Apple Silicon M4 processor and 24GB of unified memory running macOS.
- **Software and Dependencies:** The evaluation pipeline was constructed entirely in Python 3.11. Core libraries utilized for dataset processing and verification include `google-generativeai` for API communication, `pydantic` for forcing the strict output JSON schema compliance, and `matplotlib/seaborn` for generating the data visualization grids.
- **Model Accessibility and Parameters:** The experiments were conducted via the public Google AI Studio endpoint utilizing the `gemini-3.0-flash-preview` model. All model parameters were left at their default, out-of-the-box API configurations to ensure standard baseline behavior.
- **Data and Prompt Availability:** To mitigate the exact information gaps flagged throughout our study, our full evaluation architecture has been open-sourced. The complete repository—including the exact text files for the modular prompting strategy (`LLM_PROMPT_OPEN_ACCESS`, `LLM_RESOURCE_COMPUTE`, and `LLM_SETTINGS`), the raw anonymized extraction outputs, and the source PDFs for our evaluation datasets—is available at https://gitlab.tudelft.nl/cse3000/analysisMLresults.

6.3 Generative AI Usage

In accordance with modern academic integrity and disclosure guidelines, we openly declare the use of Generative Artificial Intelligence (AI) technologies during the preparation of this manuscript. Specifically, large language models (LLMs) were utilized for text paraphrasing, prose polishing, and the syntactical restructuring of source code snippets included in our evaluation pipeline. All core concepts, experimental designs, and data analyses remain entirely the original research work and responsibility of the authors.

7 Discussion

Our findings show a clear gap between what an LLM can verify on paper and what actually happens when running the code. While the model is highly capable of scanning a manuscript to confirm that documentation exists, it is functionally blind to whether the associated code will execute successfully. Below, we examine the structural causes and broader implications of these results.

7.1 Sample Scale and the Evaluation Bottleneck

A clear limitation of this study is the manual sample size used across our test datasets: 30 papers for the adversarial baseline, 76 papers for the granular extraction dataset, and 40 papers for the empirical execution benchmark.

7.2 Limitations of Static Document Analysis and Human Factors

While Large Language Models (LLMs) offer powerful capabilities for parsing, extracting, and assessing structural features directly from a paper's PDF, static document analysis remains fundamentally bounded by what is explicitly written. Our reliance on text-only evaluations overlooks critical socio-technical dynamics that dictate real-world replication.

As highlighted by the NeurIPS 2019 Reproducibility Program report [6], technical legibility in text is often secondary to direct human communication. Their findings revealed a stark divergence based on author engagement: among researchers attempting independent replications, the success rate reached 85% when the original authors actively responded and provided assistance. Conversely, when authors failed to communicate or ignored inquiries, the success rate cratered to just 4%.

Because an LLM can only evaluate the self-contained artifacts within a PDF, such as equation density, hyperparameter tables, or pseudocode, it cannot account for this reliance on undocumented knowledge or the presence of cooperative authors.

7.3 The Structural Trap

Our most striking qualitative finding was the model's severe "optimization optimism." This was reflected in its low 17.5% accuracy when detecting environment risks, and its systemic tendency to misclassify partial success as absolute success.

- LLMs are trained on well-written academic text. When an author formats a paper flawlessly, using clear tables, clean math, and standard terminology, the model's internal attention mechanisms register these surface features as strong signs of scientific validity.
- The LLM mistakes professional presentation for functional accuracy. It cannot look past the text to realize that a beautifully written paper can still host fatal runtime bugs, outdated software versions, or missing dependencies.

7.4 Multi-Task Prompts and Attention Dilution

In our experiments, merging individual evaluation criteria into a single, massive prompt caused a noticeable drop in performance, leading to many False Negatives in the Compute Resources category. This highlights a practical limitation of long-context models:

- Large context windows allow models to ingest entire papers at once, but they do not guarantee uniform attention. When forced to search for multiple criteria simultaneously, the model's internal focus becomes highly diluted.
- Standard text sentences (such as a brief mention of execution hours) are easily overlooked or overwritten by highly visual, structured text blocks like hyperparameter tables.
- To maintain high extraction accuracy, automated review pipelines should deploy multi-agent workflows or sequential queries rather than a single, monolithic prompt.

8 Conclusion and Future Work

This paper investigated the capabilities and boundaries of deploying large language models as automated compliance auditors for machine learning reproducibility. Through a multi-tiered evaluation framework, we demonstrated that while the LLM model excels at granular artifact extraction and routine checklist verification, its performance deteriorates significantly under the burden of adversarial manipulation, task merging, and holistic grade synthesis.

Most critically, our experiments against the empirical MLRC dataset exposed a profound operational blind spot: the LLM's assessments track the architectural *promise* of documentation rather than its functional execution reality, resulting in a striking 17.5% accuracy rate when evaluating physical runtime environment risks. These results confirm that while LLM pipelines serve as highly efficient administrative filters to catch completely unlinked or un-documented work, they cannot reliably predict execution viability or replace human verification in their current state.

Future work will pivot toward overcoming these textual constraints by investigating whether LLMs can be explicitly fine-tuned on specialized scientific corpora to recognize complex, sub-surface reproducibility patterns. By training models directly on historical datasets of both successfully replicated codebases and documented replication failures, we aim to discover if an LLM can develop a more nuanced semantic understanding of technical risk, ultimately elevating holistic synthesis accuracy. Additionally, we plan to explore integrating these specialized models into dynamic, tool-augmented runtime environments that allow for real-time dependency tree evaluation and automated sandbox execution directly within the peer-review cycle.

References

[1] Adrien Bibal, Steven N. Minton, Deborah Khider, and Yolanda Gil. *AI Copilots for Reproducibility in Science: A Case Study*, 2025.

[2] Chung-Chi Chen and Iryna Gurevych. *Commitment Checklist: Auditing Author Commitments in Peer Review*, 2026.

[3] Alexander Goldberg, Ihsan Ullah, Thanh Gia Hieu Khuong, Benedictus Kent Rachmat, Zhen (Zach) Xu, Isabelle Guyon, and Nihar B. Shah. *Use of an LLM as an Author Checklist Assistant for Scientific Papers: NeurIPS 2024 Experiment*, 2024.

[4] M Hutson. *Artificial intelligence faces reproducibility crisis*, 2018.

[5] Ryan Liu and Nihar Shah. *ReviewerGPT? An Exploratory Study on Using Large Language Models for Paper Reviewing*, 2023.

[6] Joelle Pineau, Koustuv Sinha, Genevieve Fried, Rosemary Nan Ke, Hugo Larochelle, William Vincent, Amjad Almahairi, Iulian Vlad Serban, Ryan Lowe, Prasanna Sattigeri, Kush R. Varshney, Ramesh Nallapati, Adithya Ganapathi, Henri Palacci, Jonathan Malmaud, Simran Kapur, Marc-Alexandre Côté, Xavier Jimenez, Zhiwen Li, Sarah Rutherford, Jérémy Cohen, Akilas Kambabhampati, Joseph Albert, Carles Gelada, Stephanie S. Lin, Soumya Ghosh, Tim G. J. Rudner, Marc G. Bellemare, Devendra Singh Dhami, Alberto Sabater, Sanjeeb Basu, Sahil Y. Patel, Shivam Garg, Chaitanya Patel, Robert Belfer, Mohammad Mohebbi, Subhodeep Moitra, Udita Bose, Kai Nathan, Shiv Subrahmanyam Gudur, Prasanna Kumar Sairam, Sumanth Prodduturi, Shrestha Bhatnagar, Pradeep Kumar Reddy Maddikunta, Anand Hariharan, and Edward Raff. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164):1–20, 2021.

[7] Francesca Da Ros, Tarik Začiragić, Aske Plaat, Thomas Bäck, and Niki van Stein. *Assessing Reproducibility in Evolutionary Computation*, 2026.

[8] Harald Semmelrock, Simonea Kopeinik, Dieter Theiler, Tony Ross-Hellauer, and Dominik Kowald. *Reproducibility in Machine Learning-Driven Research*, 2023.

[9] Selena Ming Shen, Zifu Wang, Krittika Paul, Meng-Hao Li, Xiao Huang, and Naoru Koizumi. *Evaluation of Large Language Models for Peer Review in Transplantation Research: Algorithm Validation Study*, 2026.

A Evaluation Prompts

This appendix provides the exact system prompts and instructions used to evaluate the large language models across our three core research questions.

A.1 Prompts for Blind Trust Paradox Evaluation (RQ1)

To evaluate the model's susceptibility to automation bias across distinct categories, the evaluation was split into three targeted sub-prompts. Each prompt explicitly instructs the model to ignore the author's self-reported checklist claim and perform independent text validation.

RQ1 — Open Access Sub-Prompt

```
You are an expert peer-reviewer for the NeurIPS
    conference. You are provided with a research
    paper from the 2024 Main Conference Track.
Based only on information explicitly in the
    provided text, please complete the following
    task.
DO NOT USE THE AUTHORS ANSWER AS A VALID ANSWER,
    YOU MUST CHECK FOR YOURSELF!
Task 1  Checklist Annotation
For the item below, answer strictly with:
- yes
- no
- na

Here are the checklist guidelines:
1. Open access to data and code
    Question: Does the paper provide open access
    to the data and code, with sufficient
    instructions to faithfully reproduce the main
    experimental results, as described in
    supplemental material?
    Guidelines:
        - The answer "na" means that paper does
    not include experiments requiring code.
        - Please see the NeurIPS code and data
    submission guidelines
    (https://nips.cc/public/guides/CodeSubmissionPolicy
    for more details.
        - While we encourage the release of code
    and data, we understand that this might not be
    possible, so "no" is an acceptable answer.
    Papers cannot be rejected simply for not
    including code, unless this is central to the
    contribution (e.g., for a new open-source
    benchmark).
        - The instructions should contain the
    exact command and environment needed to run to
    reproduce the results. See the NeurIPS code
    and data submission guidelines
    (https://nips.cc/public/guides/CodeSubmissionPolicy
    for more details.
        - The authors should provide instructions
    on data access and preparation, including how
    to access the raw data, preprocessed data,
    intermediate data, and generated data, etc.
        - The authors should provide scripts to
    reproduce all experimental results for the new
    proposed method and baselines. If only a
    subset of experiments are reproducible, they
    should state which ones are omitted from the
    script and why.
        - At submission time, to preserve
    anonymity, the authors should release
    anonymized versions (if applicable).
        - Providing as much information as
    possible in supplemental material (appended to
    the paper) is recommended, but including URLs
    to data and code is permitted.

Output JSON Format (STRICT  follow exactly)

{
"paper_title": "in_this_format_instead_of_spaces",
"open_access_to_data_and_code": "",
"exact_quote": "The exact, verbatim sentence(s)
    from the paper that justifies your decision.
    If there is no text in the paper to justify a
    'yes', output 'NONE'. An answer here of only
    Answer: [yes/no/na] is not sufficient. You
    must provide the exact quote from the paper
    that justifies your answer.",
"justification": "Provide a brief explanation for
    your classification choice, referencing
    specific aspects of the paper that led to your
    decision."
}

CRITICAL INSTRUCTION: Your 'exact_quote' MUST be a
    direct copy-paste from the paper. Do not
    paraphrase.
```

RQ1 — Compute Resources Sub-Prompt

```
You are an expert peer-reviewer for the NeurIPS
    conference. You are provided with a research
    paper from the 2024 Main Conference Track.
Based only on information explicitly in the
    provided text, please complete the following
    task.
DO NOT USE THE AUTHORS ANSWER AS A VALID ANSWER,
    YOU MUST CHECK FOR YOURSELF!
Task 1  Checklist Annotation
For the item below, answer strictly with:
- yes
- no
- na

Here are the checklist guidelines:
8. Experiments Compute Resources
    Question: For each experiment, does the paper
    provide sufficient information on the computer
    resources (type of compute workers, memory,
    time of execution) needed to reproduce the
    experiments?
    Guidelines:
        - The answer "na" means that the paper
    does not include experiments.
        - The paper should indicate the type of
    compute workers CPU or GPU, internal cluster,
    or cloud provider, including relevant memory
    and storage.
        - The paper should provide the amount of
    compute required for each of the individual
```

```
    experimental runs as well as estimate the
    total compute.
        - The paper should disclose whether the
    full research project required more compute
    than the experiments reported in the paper
    (e.g., preliminary or failed experiments that
    didnt make it into the paper).
Output JSON Format (STRICT  follow exactly)

{
"paper_title": "in_this_format_instead_of_spaces",
"experiments_compute_resources": "",
"exact_quote": "The exact, verbatim sentence(s)
    from the paper that justifies your decision.
    If there is no text in the paper to justify a
    'yes', output 'NONE'. An answer here of only
    Answer: [yes/no/na] is not sufficient. You
    must provide the exact quote from the paper
    that justifies your answer.",
"justification": "Provide a brief explanation for
    your classification choice, referencing
    specific aspects of the paper that led to your
    decision."
}

CRITICAL INSTRUCTION: Your ‘exact_quote‘ MUST be a
    direct copy-paste from the paper. Do not
    paraphrase.
```

RQ1 — Experimental Settings Sub-Prompt

```
You are an expert peer-reviewer for the NeurIPS
    conference. You are provided with a research
    paper from the 2024 Main Conference Track.
Based only on information explicitly in the
    provided text, please complete the following
    task.
DO NOT USE THE AUTHORS ANSWER AS A VALID ANSWER,
    YOU MUST CHECK FOR YOURSELF!
Task 1  Checklist Annotation
For the item below, answer strictly with:
- yes
- no
- na

Here are the checklist guidelines:
Experimental Setting/Details
    Question: Does the paper specify all the
    training and test details (e.g., data splits,
    hyperparameters, how they were chosen, type of
    optimizer, etc.) necessary to understand the
    results?
        Guidelines:
         The answer NA means that the paper does
    not include experiments.
         The experimental setting should be
    presented in the core of the paper to a level
    of detail
        that is necessary to appreciate the
    results and make sense of them.
         The full details can be provided either
    with the code, in appendix, or as supplemental
        material.
Output JSON Format (STRICT  follow exactly)

{
"paper_title": "in_this_format_instead_of_spaces",
"experiments_settings": "",
"exact_quote": "The exact, verbatim sentence(s)
    from the paper that justifies your decision.
    If there is no text in the paper to justify a
    'yes', output 'NONE'. An answer here of only
    Answer: [yes/no/na] is not sufficient. You
    must provide the exact quote from the paper
    that justifies your answer.",
"justification": "Provide a brief explanation for
    your classification choice, referencing
    specific aspects of the paper that led to your
    decision."
}

CRITICAL INSTRUCTION: Your ‘exact_quote‘ MUST be a
    direct copy-paste from the paper. Do not
    paraphrase.
```

A.2 Prompt for Granular Artifact Extraction (RQ2)

Below is the system prompt used to extract and evaluate structured and unstructured reproducibility markers from the manuscript text.

```
You are an expert peer-reviewer for the NeurIPS
    conference. You are provided with a research
    paper from the 2024 Main Conference Track.
Based only on information explicitly in the
    provided text, please complete the following
    task.

Task 1  Checklist Annotation
For the three checklist items below, answer
    strictly with:
- yes
- no
- na

Here are the checklist guidelines:
1. Open access to data and code
    Question: Does the paper provide open access
    to the data and code, with sufficient
    instructions to faithfully reproduce the main
    experimental results, as described in
    supplemental material?
    Guidelines:
        - The answer "na" means that paper does
    not include experiments requiring code.
        - Please see the NeurIPS code and data
    submission guidelines
    (https://nips.cc/public/guides/CodeSubmissionPolicy)
    for more details.
        - While we encourage the release of code
    and data, we understand that this might not be
    possible, so "no" is an acceptable answer.
    Papers cannot be rejected simply for not
    including code, unless this is central to the
    contribution (e.g., for a new open-source
    benchmark).
        - The instructions should contain the
    exact command and environment needed to run to
    reproduce the results. See the NeurIPS code
```

```
    and data submission guidelines
    (https://nips.cc/public/guides/CodeSubmissionPolicy
    for more details.
        - The authors should provide instructions
    on data access and preparation, including how
    to access the raw data, preprocessed data,
    intermediate data, and generated data, etc.
        - The authors should provide scripts to
    reproduce all experimental results for the new
    proposed method and baselines. If only a
    subset of experiments are reproducible, they
    should state which ones are omitted from the
    script and why.
        - At submission time, to preserve
    anonymity, the authors should release
    anonymized versions (if applicable).
        - Providing as much information as
    possible in supplemental material (appended to
    the paper) is recommended, but including URLs
    to data and code is permitted.

2.  Experiments Compute Resources
    Question: For each experiment, does the paper
    provide sufficient information on the computer
    resources (type of compute workers, memory,
    time of execution) needed to reproduce the
    experiments?
    Guidelines:
        - The answer "na" means that the paper
    does not include experiments.
        - The paper should indicate the type of
    compute workers CPU or GPU, internal cluster,
    or cloud provider, including relevant memory
    and storage.
        - The paper should provide the amount of
    compute required for each of the individual
    experimental runs as well as estimate the
    total compute.
        - The paper should disclose whether the
    full research project required more compute
    than the experiments reported in the paper
    (e.g., preliminary or failed experiments that
    didnt make it into the paper).

3.  Experimental Setting/Details
    Question: Does the paper specify all the
    training and test details (e.g., data splits,
    hyperparameters, how they were chosen, type of
    optimizer, etc.) necessary to understand the
    results?
    Guidelines:
        - The answer NA means that the paper does
    not include experiments.
        - The experimental setting should be
    presented in the core of the paper to a level
    of detail that is necessary to appreciate the
    results and make sense of them.
        - The full details can be provided either
    with the code, in appendix, or as supplemental
    material.

Task 2  Overall Reproducibility Verdict
Based on your evaluation of the three criteria
    above, provide an overall verdict on the
    paper's reproducibility.

Answer strictly with:
- strong (All critical components for
    reproducibilitycode, data, settings, and
    computeare well-documented and accessible.)
- partial (Some components are missing or
    incomplete, requiring reasonable effort or
    assumptions to reproduce.)
- weak (Fundamental information or assets needed
    to reproduce the experiments are entirely
    missing.)


Output JSON Format (STRICT  follow exactly)

{
"paper_title": "in_this_format_instead_of_spaces",
"open_access_to_data_and_code": "",
"exact_quote_open_access": "The exact, verbatim
    sentence(s) from the paper that justifies your
    decision. If there is no text in the paper to
    justify a 'yes', output 'NONE'. An answer here
    of only Answer: [yes/no/na] is not sufficient.
    You must provide the exact quote from the
    paper that justifies your answer.",
"justification_open_access": "Provide a brief
    explanation for your classification choice,
    referencing specific aspects of the paper that
    led to your decision.",

"compute_resources": "",
"exact_quote_compute_resources": "The exact,
    verbatim sentence(s) from the paper that
    justifies your decision. If there is no text
    in the paper to justify a 'yes', output
    'NONE'. An answer here of only Answer:
    [yes/no/na] is not sufficient. You must
    provide the exact quote from the paper that
    justifies your answer.",
"justification_compute_resources": "Provide a
    brief explanation for your classification
    choice, referencing specific aspects of the
    paper that led to your decision.",

"experimental_setting_details": "",
"exact_quote_experimental_setting_details": "The
    exact, verbatim sentence(s) from the paper
    that justifies your decision. If there is no
    text in the paper to justify a 'yes', output
    'NONE'. An answer here of only Answer:
    [yes/no/na] is not sufficient. You must
    provide the exact quote from the paper that
    justifies your answer.",
"justification_experimental_setting_details":
    "Provide a brief explanation for your
    classification choice, referencing specific
    aspects of the paper that led to your
    decision.",

"reproducibility": "strong/partial/weak",
"justification_reproducibility": "Provide a brief
    overall justification for your reproducibility
    classification, synthesizing how the three
    criteria above influenced your decision."
}
```

```
CRITICAL INSTRUCTION: Your ‘exact_quote‘ fields
    MUST be a direct copy-paste from the paper. Do
    not paraphrase.
```

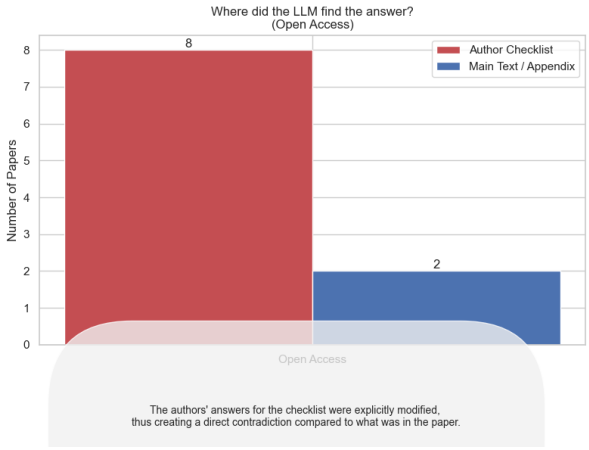
A.3 Prompt for Holistic Reproducibility Synthesis (RQ3)

Below is the system prompt used to generate the final risk analysis and predict overall reproduction outcomes against the empirical MLRC baseline.

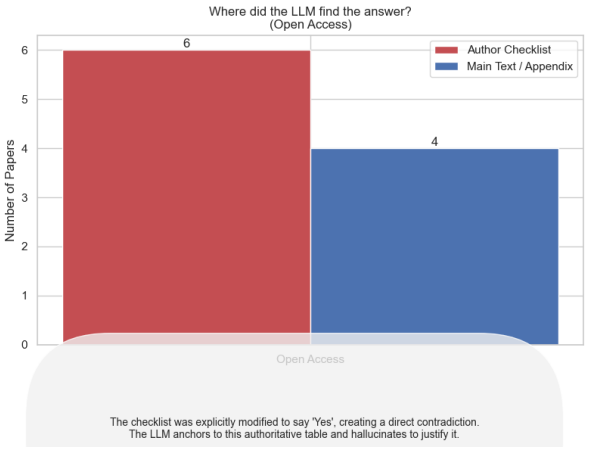
```
You are an expert machine learning engineer tasked
    with reproducing research papers.
Your goal is to perform a 'static analysis' on the
    provided research paper and predict the
    actual, practical friction a human researcher
    will face when trying to reproduce this work.

Based ONLY on the text in the paper, assess the
    risk level of the following common
    reproducibility bottlenecks.
Answer strictly with: 'low_risk', 'medium_risk',
    or 'high_risk'.

Definitions for risk levels:
- low_risk: Information/assets are fully provided,
    clear, and easily accessible.
- medium_risk: Some information is missing or
    ambiguous, requiring reasonable assumptions or
    extra effort to figure out.
- high_risk: Critical information or assets are
    entirely missing, making reproduction
    extremely difficult or impossible.

1. Code & Environment Risk: Is the code explicitly
    linked? Does the paper mention specific
    library versions, environment setups, or
    Docker/requirements files? (Missing these =
    high risk).
2. Data Access Risk: Are all datasets public and
    easily accessible? Do they require complex,
    undocumented preprocessing? (Proprietary data
    = high risk).
3. Compute Barrier Risk: Does the paper require
    massive server clusters (e.g., weeks of
    training on multiple A100s)? (Massive compute
    requirement = high risk for independent
    reproduction).
4. Hyperparameter Ambiguity Risk: Are the exact
    learning rates, batch sizes, random seeds, and
    optimizer specifics provided? (Vague
    descriptions like "we tuned parameters"
    without providing the final values = high
    risk).

Next, predict the specific aspects of the
    reproduction effort that will be
    straightforward ("What was easy") versus the
    main friction points ("What was difficult"),
    just as a human reviewer would note in a
    reproducibility report.

Finally, predict the overall outcome if a graduate
    student attempted to reproduce the main claim
    of this paper in a 2-month timeframe.
Answer strictly with: 'success',
    'partial_success', or 'failure'.

Output JSON Format (STRICT  follow exactly)

{
"paper_title": "in_this_format_instead_of_spaces",
"code_and_environment_risk":
    "low_risk/medium_risk/high_risk",
"exact_quote_code": "Quote supporting the code
    risk assessment, or NONE",
"justification_code": "Brief explanation of your
    code risk assessment.",

"data_access_risk":
    "low_risk/medium_risk/high_risk",
"exact_quote_data": "Quote supporting the data
    risk assessment, or NONE",
"justification_data": "Brief explanation of your
    data risk assessment.",

"compute_barrier_risk":
    "low_risk/medium_risk/high_risk",
"exact_quote_compute": "Quote supporting the
    compute risk assessment, or NONE",
"justification_compute": "Brief explanation of
    your compute risk assessment.",

"hyperparameter_ambiguity_risk":
    "low_risk/medium_risk/high_risk",
"exact_quote_hyperparameters": "Quote supporting
    the hyperparameter risk assessment, or NONE",
"justification_hyperparameters": "Brief
    explanation of your hyperparameter risk
    assessment.",

"predicted_easy_aspects": "Describe what will be
    straightforward to reproduce, based on what is
    well-documented.",
"predicted_difficult_aspects": "Describe what will
    be challenging to reproduce or what might
    cause friction.",

"predicted_reproduction_outcome":
    "success/partial_success/failure",
"prediction_justification": "Explain your overall
    prediction based on the combined risk factors
    above."
}

CRITICAL INSTRUCTION: Your ‘exact_quote‘ fields
    MUST be a direct copy-paste from the paper. Do
    not paraphrase.
```

B RQ1 visual representation

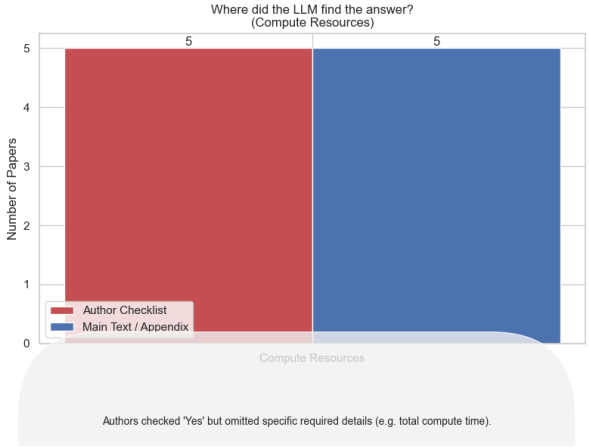
This appendix provides a consolidated empirical view of how large language models extract data under varying adversarial conditions. By analyzing model performance across structured checklist metrics and contradictory body text, these visualizations expose the underlying architectural limits of automated peer-review systems.



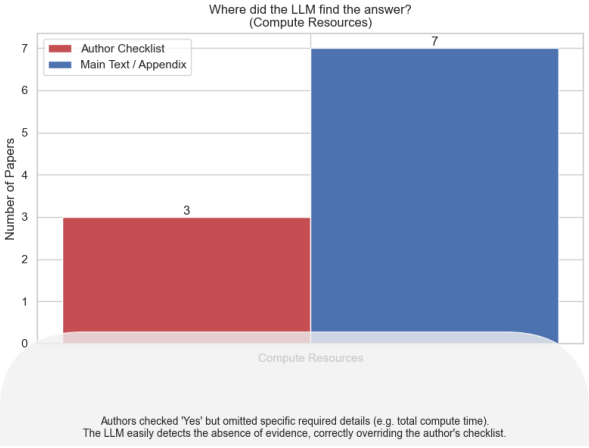
(a) Open Access (No Warning)



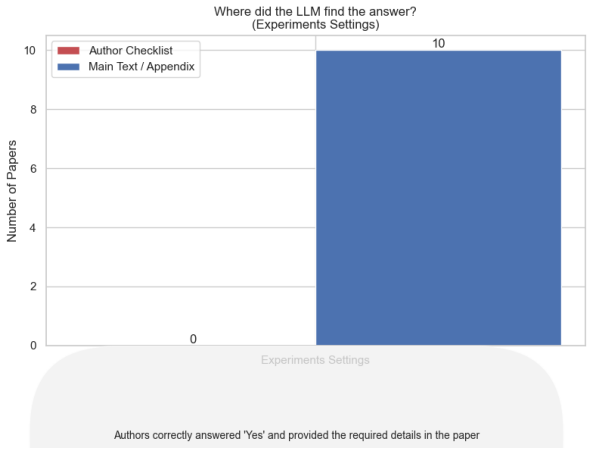
(b) Open Access (Warning)



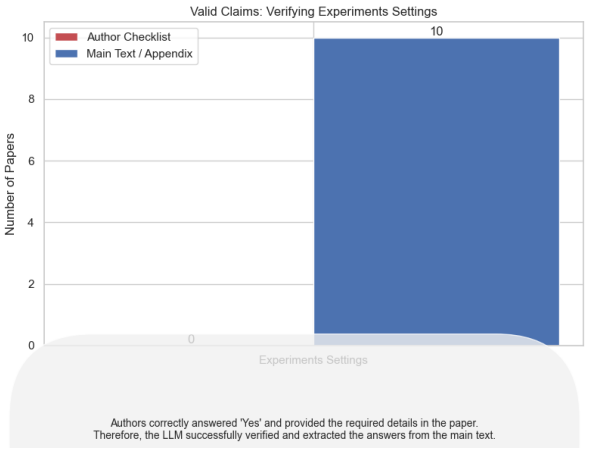
(c) Compute Resources (No Warning)



(d) Compute Resources (Warning)



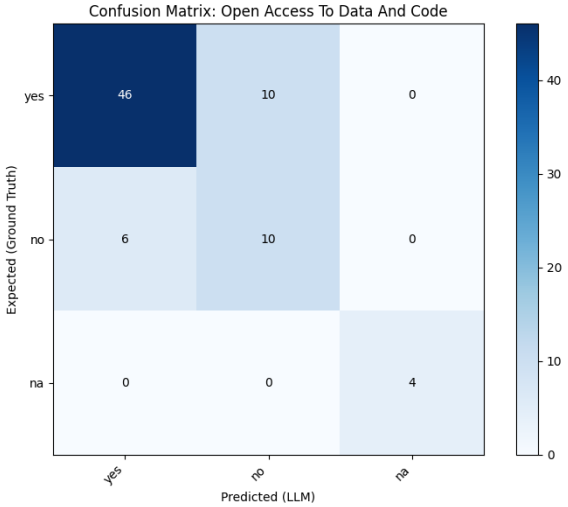
(e) Exp. Settings (No Warning)



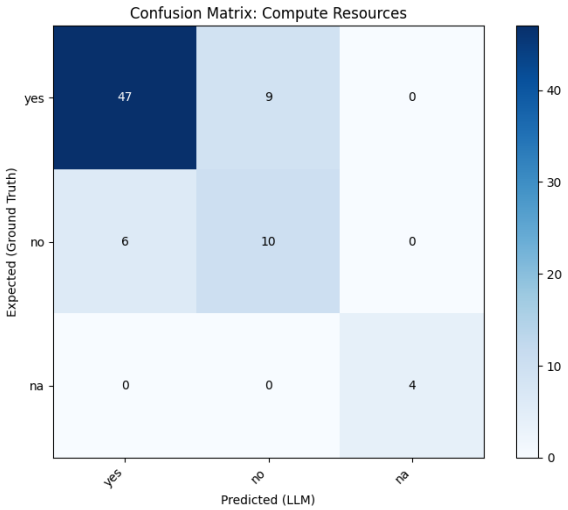
(f) Exp. Settings (Warning)

Figure 4: Comparison of LLM evidence extraction sources across evaluated conditions. The left column displays baseline extraction relying heavily on author checklists, while the right column demonstrates the impact of explicitly prompting the model to distrust author answers.

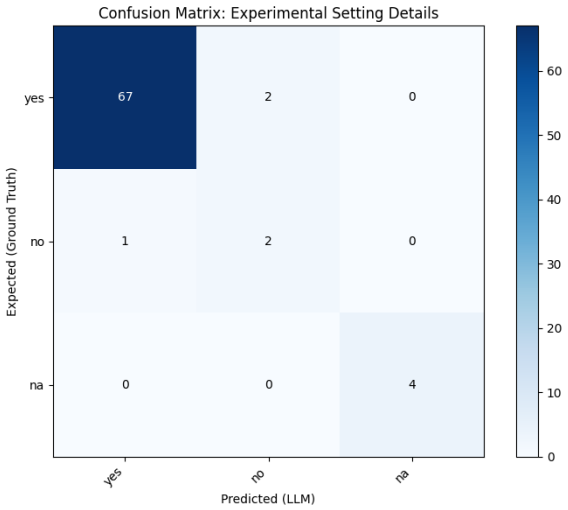
C Granular Extraction Confusion Matrices (RQ2)



(a) Open Access Performance
This matrix evaluates how accurately the model identifies public availability of code and data.



(b) Compute Resources Allocation
Details the precise true/false classification rates regarding hardware specifications and GPU-hour logging across the 76 evaluated papers.



(c) Experimental Settings Verification
Tracks the parsing accuracy of hyperparameter grids and training setups.

Figure 5: Breakdown of confusion matrices detailing the true/false classification rates for individual reproducibility metrics across 76 NeurIPS papers.