Hidden Non-Determinism in Large Language Model APIs: A Lightweight Provenance Protocol for Reproducible Generative AI Research

LUCAS ROVER*, UTFPR – Universidade Tecnológica Federal do Paraná, Brazil
YARA DE SOUZA TADANO, UTFPR – Universidade Tecnológica Federal do Paraná, Brazil

Background: Generative AI models produce non-deterministic outputs that vary across runs, even under nominally identical configurations. This variability threatens the reproducibility of studies that rely on large language model (LLM) outputs, yet most existing experiment-tracking tools were not designed for the specific challenges of text-generation workflows.

Objectives: We propose a lightweight, open-standard protocol for logging, versioning, and provenance tracking of generative AI experiments. The protocol introduces two novel documentation artifacts—Prompt Cards and Run Cards—and adopts the W3C PROV data model to create auditable, machine-readable provenance graphs linking every output to its full generation context.

Methods: We formalize the protocol and evaluate it empirically through 4,104 controlled experiments. These experiments employ nine model deployments—three locally deployed (LLaMA 3 8B, Mistral 7B, Gemma 2 9B), five closed-source API-served (GPT-4, Claude Sonnet 4.5, Gemini 2.5 Pro, DeepSeek Chat, Perplexity Sonar), and one cloud-served open-weight model (LLaMA 3 8B via Together AI, for causal isolation of infrastructure effects)—on four NLP tasks across six independent cloud providers. All models are evaluated on single-turn extraction and summarization under greedy decoding (10–30 abstracts per model). Multi-turn refinement and RAG extraction are evaluated for the three local models, Claude Sonnet 4.5, and Gemini 2.5 Pro under greedy decoding (10 abstracts each). Statistical robustness is ensured through Holm-Bonferroni correction across 68 hypothesis tests, Fisher's exact tests for binary reproducibility, bias-corrected bootstrap confidence intervals, and sensitivity analysis. We measure output variability using Exact Match Rate, Normalized Edit Distance, ROUGE-L, and BERTScore, and quantify the protocol's own overhead in terms of time and storage.

Results: Under greedy decoding (t=0), local models achieve near-perfect reproducibility (average single-turn EMR = 0.960; Gemma 2 9B: perfect 1.000 across all tasks). Closed-source API models exhibit substantial hidden non-determinism spanning a wide range (EMR 0.100–0.800), yielding a 3-fold local-vs-API gap confirmed across five providers and surviving Holm-Bonferroni correction (51/68 tests significant). Crucially, the same LLaMA 3 8B architecture served via Together AI's cloud endpoint achieves near-local reproducibility (EMR = 1.000/0.880), demonstrating that cloud deployment per se does not preclude determinism—the variability in closed-source models is consistent with infrastructure complexity rather than cloud deployment alone. Under multi-turn refinement and RAG extraction, local models maintain EMR $\geq$ 0.880, while API models exhibit near-zero reproducibility (EMR $\leq$ 0.070). The protocol adds less than 1% overhead.

Conclusions: Our results provide evidence that (1) all five API providers exhibit non-determinism under greedy decoding, while local models achieve near-perfect bitwise reproducibility; (2) API reproducibility spans a wide

*Corresponding Author.

Authors' Contact Information: Lucas Rover, ORCID: 0000-0001-6641-9224, lucasrover@utfpr.edu.br, UTFPR – Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Mecânica, Ponta Grossa, Paraná, Brazil; Yara de Souza Tadano, ORCID: 0000-0002-3975-3419, yaratadano@utfpr.edu.br, UTFPR – Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Mecânica, Ponta Grossa, Paraná, Brazil.

DOI: 10.1613/jair.1.XXXXX

range (EMR 0.010–0.800); (3) the gap extends to multi-turn and RAG regimes; (4) cloud deployment per se does not preclude reproducibility (causal isolation via Together AI); (5) temperature is the dominant user-controllable factor; and (6) provenance logging adds <1% overhead. All primary comparisons survive Holm-Bonferroni correction (51/68 significant). The protocol, implementation, and all data are publicly available.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; *Documentation*; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: reproducibility, large language models, non-determinism, provenance, generative AI, experiment tracking, W3C PROV, scientific methodology

JAIR Associate Editor: TBD

JAIR Reference Format:
Lucas Rover and Yara de Souza Tadano. 2026. Hidden Non-Determinism in Large Language Model APIs: A Lightweight Provenance Protocol for Reproducible Generative AI Research. *Journal of Artificial Intelligence Research* 0, Article 0 (February 2026), 40 pages. DOI: 10.1613/jair.1.XXXXX

1 Introduction

When a researcher queries a cloud-hosted LLM with the same prompt and temperature zero, one would reasonably expect identical outputs. Our experiments show otherwise: across five controlled seeds under greedy decoding, GPT-4 produces the same extraction result only 44% of the time, and Claude Sonnet 4.5 achieves only 19%. Meanwhile, locally deployed models such as Gemma 2 9B produce *perfectly identical* outputs every time. This hidden, provider-dependent non-determinism exemplifies a fundamental challenge introduced by the rapid adoption of large language models (LLMs) in scientific research: how to ensure that studies relying on generative AI outputs are reproducible, auditable, and scientifically rigorous. Unlike traditional computational experiments, in which deterministic algorithms produce identical results given identical inputs, LLMs exhibit inherent variability in their outputs due to stochastic sampling, floating-point non-determinism, and opaque model-versioning practices (**chen2023chatgpt**; **zhu2023reproducibility**).

Importantly, "non-reproducible" does not necessarily mean "unreliable": our experiments also show that semantic similarity (measured by BERTScore F1) remains above 0.94 across all conditions, even when exact textual match drops to zero. In other words, API outputs typically convey the same *meaning* despite differing in *phrasing*—but this distinction is invisible without systematic measurement, and many downstream analyses (meta-analyses, comparative studies, regulatory audits) require exact reproducibility.

A related subtlety concerns the `seed` parameter offered by some APIs. For API-served models, the seed parameter is advisory, not a guarantee of determinism: OpenAI explicitly documents that "determinism is not guaranteed" even when a seed is specified (**openai2024seed**), and Anthropic's Claude API does not support a seed parameter at all. Our experimental design accounts for this by treating seed variation as a control condition and measuring actual output reproducibility directly, rather than relying on API-side determinism guarantees.

This reproducibility challenge is not merely theoretical. **baker2016reproducibility** reported that over 70% of researchers have failed to reproduce another scientist's experiment, a crisis that extends to AI research (**hutson2018artificial**; **gundersen2018state**; **stodden2016enhancing**; **kapoor2023leakage**; **mukherjee2025llmreproducibility**). For generative AI specifically, the problem is compounded by several factors unique to text-generation workflows: (1) the same prompt can yield semantically similar yet textually distinct outputs across runs; (2) API-based models may undergo silent updates that alter behavior; (3) temperature and sampling parameters create a high-dimensional space of possible outputs;

and (4) no established standard exists for documenting the full context needed to understand, audit, or reproduce a generative output.

Existing experiment-tracking tools such as MLflow (**zaharia2018accelerating**), Weights & Biases (**biewald2020experiment**), and DVC (**kuprieiev2024dvc**) were designed primarily for training pipelines and numerical metrics. Although valuable for their intended purposes, these tools lack features critical for generative AI studies: structured prompt versioning, cryptographic output hashing for tamper detection, provenance graphs linking outputs to their full generation context, and environment fingerprinting specific to inference-time conditions.

Figure 1 provides an overview of the study design, protocol pipeline, and key findings. In this paper, we make three contributions, with the protocol design as the primary and most durable contribution:

(1) **A lightweight, standards-based protocol** for logging, versioning, and provenance tracking of generative AI experiments. The protocol introduces *Prompt Cards* and *Run Cards* as structured documentation artifacts, and adopts the W3C PROV data model (**w3cprov2013**) for machine-readable provenance graphs. It operationalizes—and extends to generative AI workflows—the reproducibility checklist and badge mechanisms recently adopted by JAIR (**gundersen2024improving**), providing machine-readable infrastructure that automates what those mechanisms require researchers to document manually.
(2) **A large-scale empirical case study** demonstrating both the protocol's effectiveness and the reproducibility characteristics of LLM outputs in the models and snapshots evaluated. Through 4,104 controlled experiments with nine model deployments (3 local, 5 closed-source API, 1 cloud-served open-weight for causal isolation) across four NLP tasks, 30 abstracts, six cloud providers, and five conditions (Section 4), we document a striking reproducibility gap between local and API-based inference that is invisible without systematic logging.
(3) **A reference implementation** in Python that demonstrates the protocol's practical applicability, together with all experimental data, to facilitate adoption and independent verification.

The remainder of this paper is organized as follows. Section 2 reviews related work on reproducibility in AI and experiment tracking. Section 3 formalizes the protocol design. Section 4 describes the experimental methodology. Section 5 presents the empirical results. Section 6 discusses findings, limitations, and practical implications. Section 7 concludes with directions for future work.

2 Related Work

2.1 Reproducibility in AI Research

The reproducibility crisis in AI has been documented extensively. **gundersen2018state** surveyed 400 AI papers and found that only 6% provided sufficient information for full reproducibility. **pineau2021improving** reported on the NeurIPS 2019 Reproducibility Program, which introduced reproducibility checklists and found significant gaps between reported and actual reproducibility. More recently, **gundersen2024improving** described four institutional mechanisms adopted by JAIR—reproducibility checklists, structured abstracts, badges, and reproducibility reports—establishing a community standard for what should be documented in AI research. **gundersen2018sources** identified three levels of reproducibility in AI—method, data, and experiment—and argued that all three are necessary for scientific progress. **belz2021systematic** conducted a systematic review of 601 NLP papers and confirmed pervasive under-reporting of experimental details; **belz2022quantified** further showed that missing information makes it practically impossible to assess reproducibility of human evaluations in NLP. **rogers2021changing** proposed incentive structures to improve reproducibility norms in computational linguistics. **dodge2019show** proposed improved reporting standards for ML experiments, including confidence intervals and significance tests across

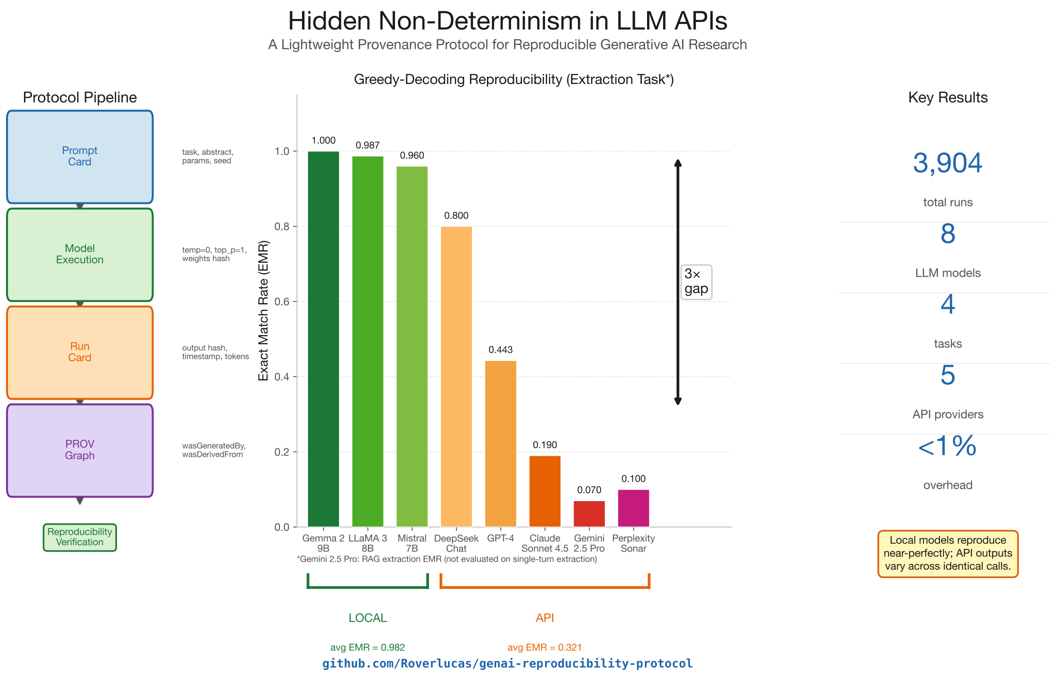

Fig. 1. Visual abstract summarizing the study design and key findings. **Left:** The provenance protocol pipeline, from Prompt Card creation through Run Card logging and W3C PROV graph generation. **Center:** Exact Match Rates (EMR) under greedy decoding for eight of the nine model deployments (Together AI's cloud LLaMA 3 8B, which mirrors local LLaMA's values, is omitted for visual clarity), illustrating the reproducibility gap between local models (green, EMR $\geq$ 0.960) and API-served models (orange/red, EMR $\leq$ 0.800). Gemini 2.5 Pro shows RAG extraction EMR (not evaluated on single-turn extraction). **Right:** Key statistics: 4,104 experiments, 9 model deployments, 4 tasks, 7 providers, <1% protocol overhead.

multiple runs, and **gundersen2022sources** provided a comprehensive taxonomy of irreproducibility sources in machine learning. More broadly, **kapoor2023leakage** identified data leakage as a widespread driver of irreproducible results across 17 scientific fields that use ML-based methods.

For generative AI specifically, **chen2023chatgpt** demonstrated that ChatGPT's outputs on NLP benchmarks exhibit non-trivial variability across identical queries, even with temperature set to zero. **zhu2023reproducibility** showed that reproducibility degrades further when tasks involve subjective judgment, such as social computing annotations. Most recently, **atil2024nondeterminism** systematically measured the non-determinism of five LLMs under supposedly deterministic settings across eight tasks, finding accuracy variations up to 15% across runs and introducing the Total Agreement Rate (TAR) metric. **ouyang2024nondeterminism** confirmed that temperature zero does not guarantee determinism in ChatGPT code generation. Concurrently, **yuan2025nondeterminism** traced such non-determinism to numerical precision issues in GPU kernels and proposed LayerCast as a mitigation strategy—a hardware-level fix that reduces but does not eliminate non-determinism, and that is not available to researchers using closed API services. The PyTorch documentation (**pytorch2024deterministic**) further catalogues sources of non-determinism in GPU operations, providing the `torch.use_deterministic_algorithms()`

flag as a partial mitigation for training; however, this flag is unavailable for API-served inference. Our Exact Match Rate (EMR) metric is closely related to **atil2024nondeterminism**'s Total Agreement Rate (TAR), which measures the fraction of runs producing the modal output; EMR instead measures the fraction of *all output pairs* that match exactly, providing a more sensitive measure when agreement is low and no clear modal output exists. Our work complements these studies in four specific ways. First, whereas prior studies (including Atil et al.'s five-model, eight-task study) measure variability post hoc, we provide a structured provenance protocol that enables *prospective* documentation and audit—answering not only "how much variability?" but also "why did these outputs differ?" through cryptographic hashing and W3C PROV graphs. Second, we directly compare local and API-based inference on identical tasks with identical prompts across nine model deployments and six independent cloud providers, isolating the deployment paradigm as a variable—including a causal isolation test with the same open-weight model under local and cloud deployment—and confirming that API non-determinism is a consistent pattern across providers (EMR 0.010–0.800). Third, we extend beyond single-turn evaluation to include multi-turn refinement and retrieval-augmented generation, demonstrating that reproducibility characteristics generalize across interaction regimes. Fourth, we quantify the overhead of systematic logging, demonstrating that the "cost of knowing" is negligible.

2.2 Experiment Tracking Tools

Several tools exist for tracking machine learning experiments, although none was designed specifically for generative AI text-output workflows:

MLflow (**zaharia2018accelerating**) provides experiment tracking, model packaging, and deployment. It logs parameters, metrics, and artifacts, but focuses on training pipelines and numerical outcomes rather than text-generation provenance.

Weights & Biases (**biewald2020experiment**) offers experiment tracking with visualization dashboards. It supports prompt logging but lacks structured prompt versioning, cryptographic output hashing, and provenance graph generation.

DVC (**kuprieiev2024dvc**) provides data versioning through git-like operations. While effective for dataset management, it does not address run-level provenance or prompt documentation.

OpenAI Evals (**openai2023evals**) is a framework for evaluating LLM outputs against benchmarks. It provides structured evaluation but is tightly coupled to OpenAI's ecosystem and does not generate interoperable provenance records.

LangSmith (**langsmith2023**) offers tracing and evaluation for LLM applications. It captures detailed execution traces but uses a proprietary format and requires cloud connectivity.

More broadly, **bommasani2022opportunities** identified reproducibility as a key risk for foundation models, and **liang2023holistic** proposed the HELM benchmark for holistic evaluation of language models, including robustness and fairness dimensions that complement our reproducibility focus. In the provenance space, **padovani2025yprov** recently introduced yProv4ML, a framework that captures ML provenance in PROV-JSON format with minimal code modifications; our protocol shares the commitment to W3C PROV and SHA-256 hashing but differs in three key respects: (i) we target inference-time stochastic text generation rather than training pipelines; (ii) our Run Cards capture prompt-level metadata (prompt hash, seed status, interaction regime) not present in training-oriented schemas; and (iii) we provide empirical evidence quantifying why such logging is necessary for API-served models.

Table 1 provides a systematic feature-by-feature comparison of our protocol with these tools. The key distinction is not merely one of tooling but of *scientific capability*: existing tools log what happened during training (parameters, metrics, artifacts), whereas our protocol enables answering questions that these tools

Table 1. Comparison of our protocol with existing reproducibility tools and frameworks for GenAI experiments. Checkmarks (✓) indicate full support; tildes (∼) indicate partial support; dashes (–) indicate no support.

Feature	Ours	MLflow	W&B	DVC	OpenAI Evals	LangSmith
Prompt versioning (Prompt Card)	✓	–	∼	–	∼	∼
Run-level provenance (W3C PROV)	✓	–	–	–	–	–
Cryptographic output hashing	✓	–	–	✓	–	–
Seed & param logging	✓	✓	✓	–	✓	✓
Environment fingerprinting	✓	∼	∼	∼	–	–
Model weights hashing	✓	–	∼	✓	–	–
Overhead <1% of inference	✓	∼	∼	N/A	N/A	∼
Designed for GenAI text output	✓	–	–	–	✓	✓
Open standard (PROV-JSON)	✓	–	–	–	–	–
Local-first (no cloud dependency)	✓	✓	–	✓	–	–

cannot—specifically, whether two generative outputs are provably derived from identical configurations, which exact factor caused a divergence between non-identical outputs, and whether an output has been tampered with post-generation. These capabilities require the combination of cryptographic hashing, structured prompt documentation, and W3C PROV provenance graphs that no existing tool provides. In short, our contribution is not an alternative experiment tracker but a *reproducibility assessment framework* designed for the unique challenges of stochastic text generation.

2.3 Provenance in Scientific Computing

Data provenance—the lineage of data through transformations—has a rich history in database systems and scientific workflows (**herschel2017survey**). The W3C PROV family of specifications (**w3cprov2013**) provides a standardized data model for representing provenance as directed acyclic graphs of *entities*, *activities*, and *agents*. **samuel2022computational** applied provenance tracking to computational biology workflows, demonstrating its value for reproducibility. However, to our knowledge, no prior work has applied W3C PROV specifically to generative AI experiment workflows, in which the challenge involves not only tracking data lineage but also capturing the stochastic generation context that determines output variability.

Taken together, these gaps point to a clear need: a lightweight, standards-based protocol that bridges generative AI inference with the provenance infrastructure already established in scientific computing. The next section presents our design for such a protocol.

3 Protocol Design

Our protocol addresses the question: *What is the minimum set of metadata that must be captured for each generative AI run to enable auditing, reproducibility assessment, and provenance tracking?* We address this question through four complementary components.

3.1 Scope and Design Principles

The protocol is designed around three principles:

(1) **Completeness**: Every factor that can influence a generative output must be captured—prompt text, model identity and version, inference parameters, environment state, and timestamps.

(2) **Negligible overhead**: The logging process must not materially affect the experiment. We target <1% overhead relative to inference time.
(3) **Interoperability**: All artifacts are stored in open, machine-readable formats (JSON, PROV-JSON), aligned with the FAIR (Findable, Accessible, Interoperable, Reusable) principles (**wilkinson2016fair**), to enable tool integration and long-term preservation.

3.2 Prompt Cards

A *Prompt Card* is a versioned documentation artifact that captures the design rationale and metadata for a prompt template used in experiments. Each Prompt Card contains:

- `prompt_id`: Unique identifier
- `prompt_hash`: SHA-256 hash of the prompt text, enabling tamper detection
- `version`: Semantic version number
- `task_category`: Classification of the task (e.g., summarization, extraction)
- `objective`: Natural-language description of what the prompt is designed to achieve
- `assumptions`: Explicit assumptions about inputs and expected behavior
- `limitations`: Known limitations or failure modes
- `target_models`: Models for which the prompt was designed and tested
- `expected_output_format`: Description of the expected output structure
- `interaction_regime`: Single-turn, multi-turn, or chain-of-thought
- `change_log`: History of modifications

Prompt Cards serve two purposes: they document design intent (supporting human understanding) and they provide a citable, hashable reference for automated provenance tracking. The concept draws inspiration from Model Cards (**mitchell2019model**), Datasheets for Datasets (**gebru2021datasheets**), and model info sheets for reproducibility assessment (**kapoor2023leakage**), extending the structured-documentation paradigm to the prompt layer of the generative AI pipeline.

3.3 Run Cards

A *Run Card* captures the complete execution context of a single generative AI run. Each Run Card records 24 core fields organized into five groups (the complete JSON schema in Appendix B includes these fields plus additional metadata such as `researcher_id`, `affiliation`, `system_logs`, and `errors`):

(1) **Identification**: `run_id`, `task_id`, `task_category`, `prompt_hash`, `prompt_text`
(2) **Model context**: `model_name`, `model_version`, `weights_hash`, `model_source`
(3) **Parameters**: `inference_params` (temperature, top_p, top_k, max_tokens, seed, decoding_strategy), `params_hash`
(4) **Input/Output**: `input_text`, `input_hash`, `output_text`, `output_hash`, `output_metrics`
(5) **Execution metadata**: `environment` (OS, architecture, Python version, hostname), `environment_hash`, `code_commit`, timestamps (start/end), `execution_duration_ms`, `logging_overhead_ms`, `storage_kb`

For API-served models, optional extension fields capture provider-specific metadata that may help diagnose non-determinism: `api_request_id`, `api_response_headers`, `api_model_version_returned`, `api_region`, and a `seed_status` field that distinguishes between seeds that were "sent" to the API, "logged-only" (recorded for protocol parity but not sent, as with Claude), or "not-supported" by the provider. This formalization ensures that the advisory or absent nature of API seed parameters is captured as structured metadata rather than left as an undocumented assumption.

Figure 2 illustrates the Run Card schema as a minimal structured record.

Run Card Schema (24 core + extension fields)
1. Identification
`run_id` · `task_id` · `task_category` · `prompt_hash` · `prompt_text`
2. Model Context
`model_name` · `model_version` · `weights_hash` · `model_source`
3. Parameters
`inference_params` {temp, top_p, top_k, max_tokens, seed, strategy} · `params_hash`
4. Input/Output
`input_text` · `input_hash` · `output_text` · `output_hash` · `output_metrics`
5. Execution Metadata
`environment` · `environment_hash` · `code_commit` · timestamps · `duration_ms` · `overhead_ms` · `storage_kb`
API Extensions (optional)
`api_request_id` · `api_region` · `seed_status` ∈ {sent, logged-only, not-supported}
Workflow Extensions (optional)
`conversation_history_hash` · `turn_index` · `retrieval_context_hash` · `parent_run_id`

Fig. 2. Run Card minimal schema. All SHA-256 hashes (5 total) enable tamper detection and automated comparison. API and workflow extension fields are optional.

The separation of logging overhead from execution time is deliberate: it allows researchers to verify that the protocol itself does not confound experimental measurements.

3.3.1 Normative Field Requirements. To support adoption as a citable specification, we classify Run Card fields using normative language following RFC 2119 (**rfc2119**):

- **MUST** (required for audit completeness): `run_id`, `prompt_text`, `prompt_hash`, `model_name`, `model_version`, `output_text`, `output_hash`, `timestamp_start`, `inference_params` (including temperature, seed, decoding strategy).
- **SHOULD** (strongly recommended): `input_hash`, `params_hash`, `environment_hash`, `weights_hash` (local), `code_commit`, `execution_duration_ms`, `logging_overhead_ms`, `seed_status` (API).
- **MAY** (optional, context-dependent): `api_request_id`, `api_response_headers`, `api_region`, `conversation_history_hash`, `turn_index`, `retrieval_context_hash`, `parent_run_id`, `researcher_id`, `affiliation`.

A conforming implementation MUST populate all MUST fields and SHOULD populate all SHOULD fields. The MUST set is minimal: removing any MUST field renders at least one audit question from Section 6.8 unanswerable.

3.4 W3C PROV Integration

Each experimental group (defined by a unique model–task–condition–abstract combination) is automatically translated into a W3C PROV-JSON document (**w3cprov2013**) that expresses the generation provenance as a directed graph. The mapping defines:

- **Entities**: Prompt, InputText, ModelVersion, InferenceParameters, Output, ExecutionMetadata
- **Activities**: RunGeneration (the inference execution)
- **Agents**: Researcher, SystemExecutor (the execution environment)

PROV relations capture the causal structure:

- `used`: RunGeneration used Prompt, InputText, ModelVersion, InferenceParameters
- `wasGeneratedBy`: Output wasGeneratedBy RunGeneration
- `wasAssociatedWith`: RunGeneration wasAssociatedWith Researcher, SystemExecutor
- `wasAttributedTo`: Output wasAttributedTo Researcher

- `wasDerivedFrom`: Output wasDerivedFrom InputText

This standardized representation enables automated reasoning about experiment provenance, including detecting when two runs share identical configurations and identifying the specific factors that differ between non-identical outputs. The choice of W3C PROV over plain JSON logs is deliberate: PROV's formal semantics allow automated tools to traverse the provenance graph and answer queries such as "what changed between these two runs?" without custom parsing logic. An abbreviated example document is given in Appendix C; to illustrate the structure concisely, the core provenance chain is:

$\texttt{Prompt} \longrightarrow^{\text{used}} \textbf{RunGeneration} \longrightarrow^{\text{generated}} \texttt{Output}$
$\texttt{InputText} \longrightarrow^{\text{used}} \textbf{RunGeneration} \longrightarrow^{\text{assoc.}} \texttt{Researcher}$
$\texttt{ModelVersion} \longrightarrow^{\text{used}} \textbf{RunGeneration}; \quad \texttt{Output} \longrightarrow^{\text{derived}} \texttt{InputText}$

3.5 Reproducibility Checklist

We provide a 15-item checklist organized into four categories—Prompt Documentation, Model and Environment, Execution and Output, and Provenance—that researchers can use to self-assess the reproducibility of their generative AI studies. The complete checklist is provided in Appendix A.

3.6 Extensions for Advanced Workflows

The protocol's field schema accommodates complex workflows through optional extension fields. Our empirical evaluation exercises two of these extensions—multi-turn dialogues and RAG—while the remaining extensions are specified in the reference implementation's schema:

- **Multi-turn dialogues:** A `conversation_history_hash` field and `turn_index` enable linking each turn to the full conversation state. *Evaluated in Task 3 (multi-turn refinement) using Ollama's `/api/chat` endpoint.*
- **RAG:** Fields for retrieval context (with hashes) trace which external information influenced the output. *Evaluated in Task 4 (RAG extraction) with prepended context passages.*
- **Tool use and function calling:** Fields for available tools, tool calls (with arguments, results, and hashes) capture the full tool-use chain.
- **Chain-of-thought / agent workflows:** A `parent_run_id` field supports hierarchical provenance graphs for multi-step reasoning chains.

3.7 Formal Definition and Audit Completeness

We define the protocol as a tuple $\mathcal{P} = (PC, RC, G, CL)$, where PC is a Prompt Card, RC is a Run Card, G is a W3C PROV graph, and CL is the reproducibility checklist. Each Run Card RC_i is itself a tuple of field groups: $RC_i = (Id, Mod, Par, IO, Env, H)$, where H denotes the set of five SHA-256 hashes (prompt, input, parameters, environment, output).

We define an *audit question* as a predicate Q over one or more Run Cards. The protocol satisfies the following *audit completeness* property: for a set of 10 audit questions $\{Q_1, \ldots, Q_{10}\}$ (defined in Section 6.8), every Q_j is answerable if and only if all field groups are populated. Formally:

$$\forall Q_j \in \{Q_1, \ldots, Q_{10}\} : \text{answerable}(Q_j, RC_i) \Leftrightarrow \bigwedge_{g \in \text{required}(Q_j)} g \subseteq RC_i \tag{1}$$

where required(Q_j) maps each question to its minimal set of required field groups. The ablation analysis in Section 6.8 demonstrates that every field group is in the required set of at least one question, establishing protocol *minimality*: removing any field group violates Equation 1 for at least one Q_j.

The *differential diagnosis* property follows from the hash fields: given two Run Cards RC_a, RC_b with $H^a_{\text{output}} \neq H^b_{\text{output}}$, the protocol enables automatic identification of the divergence source by comparing the remaining hashes. If $H^a_{\text{prompt}} = H^b_{\text{prompt}}$, $H^a_{\text{input}} = H^b_{\text{input}}$, $H^a_{\text{params}} = H^b_{\text{params}}$, and $H^a_{\text{env}} = H^b_{\text{env}}$, then the output difference is attributable to non-determinism in the generation process itself—precisely the phenomenon we measure empirically in Section 5.

Having defined the protocol's components and formal properties, we now evaluate it empirically along two dimensions: the reproducibility characteristics it reveals across different models and conditions, and the overhead it imposes on the experimental workflow.

4 Experimental Setup

We designed a controlled experiment to simultaneously evaluate (a) the reproducibility characteristics of LLM outputs under varying conditions and (b) the overhead imposed by our logging protocol.

4.1 Models and Infrastructure

We evaluate nine model deployments representing three deployment paradigms: three locally deployed open-weight models, five cloud API-served proprietary or closed models, and one open-weight model served via a cloud API (for causal isolation of infrastructure effects). All local models were served through Ollama v0.15.5 (**ollama2024**) on an Apple M4 system with 24 GB unified memory running macOS 14.6 with Python 3.14.3. API-served models span six independent providers: OpenAI (GPT-4), Anthropic (Claude Sonnet 4.5), Google (Gemini 2.5 Pro), DeepSeek (DeepSeek Chat), Perplexity (Sonar, an online model with search augmentation), and Together AI (LLaMA 3 8B, the same architecture as our local deployment).

4.1.1 Local Models. **LLaMA 3 8B** (**grattafiori2024llama3**): An open-weight model in Q4_0 quantization. Local deployment provides complete control over the execution environment, eliminating confounding factors such as network latency, server-side batching, and silent model updates. The model's SHA-256 weights hash was recorded per run via the Ollama API.

Mistral 7B (**jiang2023mistral**): An open-weight model (Q4_0 quantization) with a sliding-window attention mechanism, providing a second data point for local inference reproducibility at a similar parameter scale.

Gemma 2 9B (**team2024gemma**): Google's open-weight model (Q4_0 quantization), representing a third local model from an independent model family. Gemma 2 proved to be the most deterministic model in our study, though its inference time is substantially higher than the other local models (∼180 s per run vs. 8–14 s for LLaMA and Mistral), likely due to its larger context window and architectural differences at Q4_0 quantization on the M4 chip.

4.1.2 API-Served Models. **GPT-4** (**achiam2023gpt4**): Accessed via the OpenAI API (`openai` Python SDK v1.59.9) with controlled seed parameters. The API returned `gpt-4-0613` as the resolved model version in all runs. The API introduces additional sources of variability: load balancing, server-side batching, potential model-version updates, and floating-point non-determinism across different hardware.

Claude Sonnet 4.5 (**anthropic2024claude**): Accessed via the Anthropic API using a lightweight `urllib`-based runner (no SDK dependency). Claude's API does not support a `seed` parameter; we set `temperature=0` for greedy decoding and logged a seed value for protocol parity (marked as `logged-only-not-sent-to-api`). This provides an independent replication of the API non-determinism phenomenon on a second provider (Anthropic), complementing the OpenAI evaluation.

Gemini 2.5 Pro (**reid2024gemini15**): Accessed via the Google AI Studio REST API using a lightweight `urllib`-based runner (no SDK dependency). Gemini's API supports a `seed` parameter in the generation configuration; we set `seed=42` and `temperature=0` for greedy decoding. Gemini 2.5 Pro is a "thinking" model that uses internal reasoning tokens before producing output; the `maxOutputTokens` budget was set to 8,192 to accommodate this overhead. This model provides a second independent API model for multi-turn and RAG experiments (alongside Claude Sonnet 4.5), enabling cross-provider validation of the multi-turn reproducibility gap. Evaluated under C1 on Tasks 3–4 (10 abstracts, 100 runs).

DeepSeek Chat (DeepSeek): Accessed via the OpenAI-compatible API. DeepSeek Chat represents a fourth independent API provider, allowing us to test whether the non-determinism pattern generalizes beyond OpenAI, Anthropic, and Google. No seed parameter is supported; we set `temperature=0` for greedy decoding. DeepSeek Chat is evaluated under condition C1 on single-turn tasks only (10 abstracts, 100 runs).

Perplexity Sonar (Perplexity): An online, search-augmented language model accessed via the Perplexity API. Unlike the other models, Sonar incorporates real-time web search results into its generation context, introducing an additional source of variability beyond model-internal non-determinism. This model represents a fifth independent API provider and a qualitatively different deployment paradigm (search-augmented generation). Evaluated under C1 on single-turn tasks only (10 abstracts, 100 runs).

LLaMA 3 8B via Together AI (Together AI): The same LLaMA 3 8B architecture as our locally deployed model, but served via Together AI's cloud inference endpoint (INT4-quantized "Lite" variant). This deployment provides a critical causal isolation test: by running the *same model architecture* under both local and cloud-served conditions with identical prompts, seeds, and temperature, any reproducibility difference can be attributed to infrastructure rather than model architecture. Accessed via the OpenAI-compatible Together AI API with a lightweight `urllib`-based runner. Evaluated under C1 and C2 on single-turn tasks (10 abstracts, 200 runs).

4.2 Tasks

We evaluate four tasks that span the output-structure spectrum and interaction complexity:

Task 1: Scientific Summarization. Given a scientific abstract, produce a concise summary in exactly three sentences covering the main contribution, methodology, and key quantitative result. This is an open-ended generation task in which the model has considerable freedom in word choice and phrasing.

Task 2: Structured Extraction. Given a scientific abstract, extract five fields (objective, method, key_result, model_or_system, benchmark) into a JSON object. This is a constrained generation task in which the output format is fixed and the model must select, rather than generate, content.

Task 3: Multi-turn Refinement. A three-turn dialogue in which the model first extracts structured information, then receives feedback requesting more detail, and finally produces a refined extraction. This tests reproducibility under conversational state accumulation, using Ollama's `/api/chat` endpoint for local models.

Task 4: RAG Extraction. The same structured extraction task as Task 2, but with an additional retrieved context passage prepended to the input. This tests whether augmenting the prompt with external context affects reproducibility.

4.3 Input Data

We use 30 widely-cited scientific abstracts from landmark AI/ML papers, including **vaswani2017attention** (Transformer), **devlin2019bert** (BERT), **brown2020language** (GPT-3), **raffel2020exploring** (T5),

Table 2. Experimental design: conditions, parameters, and expected outcomes.

Cond.	Description	Temp.	Seed	Reps	Expected Outcome
C1	Fixed seed, greedy	0.0	42 (fixed)	5	Deterministic output
C2	Variable seeds, greedy	0.0	5 different	5	Near-deterministic
$C3_{t=0.0}$	Temp. baseline	0.0	per-rep	3	Deterministic
$C3_{t=0.3}$	Low temperature	0.3	per-rep	3	Low variability
$C3_{t=0.7}$	High temperature	0.7	per-rep	3	High variability

Note: Tasks 1–2 are evaluated under all five conditions (C1, C2, C3) for the five models evaluated under all three condition types (LLaMA 3, Mistral 7B, Gemma 2 9B, GPT-4, and Claude Sonnet 4.5); GPT-4 C1 coverage is partial due to quota exhaustion—C2 is used as GPT-4's primary greedy condition (see Section 6.7), and under C1 only for DeepSeek Chat and Perplexity Sonar. Tasks 3–4 (multi-turn, RAG) are evaluated under C1 only for the three local models, Claude Sonnet 4.5, and Gemini 2.5 Pro. Total: 4,104 logged runs across 9 model deployments (including 200 chat-format control runs for LLaMA 3). For API-served models, C2 uses the same fixed seed as C1; the seed parameter is advisory and does not guarantee determinism.

wei2022chain (Chain-of-Thought), as well as seminal works on GANs, ResNets, VAEs, LSTMs, CLIP, DALL-E 2, Stable Diffusion, LLaMA, InstructGPT, PaLM, and others. These abstracts vary in length (74–227 words), technical complexity, and the number of quantitative results reported, thereby providing substantial diversity in the generation challenge.

4.4 Experimental Conditions

We define five conditions (Table 2) that systematically vary the factors hypothesized to affect reproducibility:

Design principle for API models. For cloud-hosted APIs whose `seed` parameter is advisory rather than deterministic (as documented by OpenAI for GPT-4 (**openai2024seed**)) or entirely absent (as with Claude), the fixed-vs.-variable seed distinction has no guaranteed effect server-side. We therefore treat C2 as the primary test of determinism under greedy decoding for such models.

C1 (Fixed seed, greedy decoding): Temperature = 0, seed = 42 for all 5 repetitions. This represents the maximum-control condition and should yield deterministic outputs.

C2 (Variable seeds, greedy decoding): Temperature = 0, seeds = {42, 123, 456, 789, 1024}. This condition tests whether seed variation affects outputs when greedy decoding is used.

C3 (Temperature sweep): Three sub-conditions at $t \in \{0.0, 0.3, 0.7\}$ with 3 repetitions each, using different seeds per repetition. This condition characterizes how temperature affects output variability.

Run counts. For Tasks 1–2 (extraction and summarization), each of the five models with full condition coverage is evaluated under C1 (5 runs), C2 (5 runs), and C3 (9 runs = 3 temperatures × 3 reps) per abstract. LLaMA 3 uses 30 abstracts (1,140 runs); the newer models (Mistral 7B, Gemma 2 9B, Claude Sonnet 4.5) use 10 abstracts (380 runs each). For GPT-4, quota exhaustion limited collection to 724 runs (C2: 300/300; C3: 416/450; C1: 8/300 excluded). DeepSeek Chat and Perplexity Sonar are evaluated under C1 with 10 abstracts × 5 reps × 2 tasks = 100 runs each (200 runs total). For Tasks 3–4 (multi-turn and RAG), the three local models, Claude Sonnet 4.5, and Gemini 2.5 Pro are evaluated under C1 with 10 abstracts × 5 repetitions = 50 runs each (500 runs total). Together AI (LLaMA 3 8B) is evaluated under C1 and C2 with 10 abstracts × 5 reps × 2 tasks × 2 conditions = 200 runs. **Grand total: 4,104 valid runs.**

Table 3 summarizes the per-model run distribution.

[1]One Claude run (0.03%) returned an empty output due to API timeout and is excluded from variability metrics.

Table 3. Run distribution across models and tasks.

Model	Tasks 1–2	Tasks 3–4	Total
LLaMA 3 8B	1,140	100	1,240
Mistral 7B	380	100	480
Gemma 2 9B	380	100	480
GPT-4	724	—	724
Claude Sonnet 4.5	380	100	480
Gemini 2.5 Pro	—	100	100
DeepSeek Chat	100	—	100
Perplexity Sonar	100	—	100
Together AI (LLaMA 3 8B)	200	—	200
Chat-format control†	200	—	200
Total	**3,604**	**500**	**4,104**[1]

†LLaMA 3 8B via `/api/chat` endpoint (Appendix H).

4.5 Metrics

We adopt an operational definition of reproducibility at three levels, each mapped to a specific metric:

- **Exact reproducibility** (string-level): Two outputs are identical character-by-character. Measured by *Exact Match Rate (EMR)*.
- **Near reproducibility** (edit-level): Two outputs differ only in minor surface variations (punctuation, whitespace, synonym substitution). Measured by *Normalized Edit Distance (NED)*.
- **Semantic reproducibility** (meaning-level): Two outputs convey the same information despite different phrasing. Measured by *ROUGE-L F1* and *BERTScore F1*.

This three-level framework allows us to distinguish between outputs that are bitwise identical (EMR $= 1$), textually close (NED < 0.05), and semantically equivalent (ROUGE-L > 0.90). All variability metrics are computed over all $\binom{n}{2}$ unique output pairs within each experimental group (defined by model, task, condition, and abstract):

Exact Match Rate (EMR): The fraction of output pairs that are character-for-character identical. EMR $= 1.0$ indicates perfect reproducibility; EMR $= 0.0$ indicates that no two outputs match exactly. With $n = 5$ repetitions per group ($\binom{5}{2} = 10$ pairs), per-abstract EMR values are discrete: $\{0.0, 0.1, \ldots, 1.0\}$; with $n = 3$ (C3 conditions), EMR takes values in $\{0.0, 0.333, 0.667, 1.0\}$. This granularity should be considered when interpreting standard deviations and confidence intervals for small sample sizes.

Normalized Edit Distance (NED): The Levenshtein edit distance (**levenshtein1966binary**) between each pair, normalized by the length of the longer string. NED $= 0.0$ indicates identical outputs; higher values indicate greater textual divergence.

ROUGE-L F1: The F1 score based on the longest common subsequence at the word level (**lin2004rouge**). This captures semantic similarity even when surface forms differ. ROUGE-L $= 1.0$ indicates identical word sequences.

Our primary metrics (EMR, NED, ROUGE-L) focus on exact and near reproducibility, which are the most direct measures for our research question. To complement these surface-level metrics, we also compute **BERTScore F1** (**zhang2020bertscore**)—an embedding-based semantic similarity metric—for all conditions. BERTScore captures meaning-level equivalence that surface metrics may miss (e.g.,

Table 4. Exact Match Rate (EMR) under greedy decoding (t=0) across five models and two single-turn tasks, with 95% bootstrap confidence intervals (n_{boot}=10,000). For local models, values reflect condition C1 (fixed seed); for GPT-4, C2 (variable-seed greedy, as C1 has insufficient coverage); for Claude, C1 (Claude's API does not support a seed parameter). Higher is more reproducible.

Model	Source	Extraction EMR	Summarization EMR
Gemma 2 9B	Local	1.000 [1.00, 1.00]	1.000 [1.00, 1.00]
LLaMA 3 8B	Local	0.987 [0.96, 1.00]	0.947 [0.89, 0.99]
Mistral 7B	Local	0.960 [0.88, 1.00]	0.840 [0.72, 0.96]
GPT-4	API	0.443 [0.32, 0.57]	0.230 [0.16, 0.30]
Claude Sonnet 4.5	API	0.190 [0.05, 0.40]	0.020 [0.00, 0.05]

paraphrases), providing a fourth perspective on reproducibility. For the structured extraction task, we additionally report **JSON validity rate**, **schema compliance rate**, and **field-level accuracy**, which measure whether outputs are syntactically valid JSON, contain all expected fields, and agree on individual field values across runs, respectively (see Appendix D for detailed results).

For protocol overhead, we measure:

- **Logging time**: Wall-clock time spent on hashing, metadata collection, and file I/O, measured separately from inference time.
- **Storage**: Size of each run record (JSON) and total storage for all protocol artifacts.
- **Overhead ratio**: Logging time as a percentage of total execution time.

All EMR values in Tables 4 and 8 are accompanied by 95% bootstrap confidence intervals (10,000 resamples over per-abstract EMR values, percentile method).

5 Results

5.1 Reproducibility Under Greedy Decoding

Table 4 presents Exact Match Rates under greedy decoding for the five models with full condition coverage; results for DeepSeek Chat, Perplexity Sonar, and Gemini 2.5 Pro are reported in the text below. Table 5 provides the full three-level reproducibility assessment.

5.1.1 Local Models: Near-Perfect to Perfect Reproducibility. **Finding 1: Gemma 2 9B achieves perfect bitwise reproducibility under greedy decoding.** Across all tasks and conditions with t=0, Gemma 2 9B produces EMR = 1.000 with NED = 0.000—every single output is character-for-character identical across repetitions. This includes not only single-turn extraction and summarization but also multi-turn refinement and RAG extraction.

Finding 2: All three local models achieve high reproducibility. LLaMA 3 8B attains EMR = 0.987 for extraction and 0.947 for summarization; Mistral 7B achieves 0.960 and 0.840, respectively. The small deviations from perfect reproducibility in LLaMA 3 and Mistral 7B are *entirely attributable* to a cold-start effect on the first inference call after model loading. A post-hoc analysis of all non-unanimous abstract groups (i.e., groups where at least one repetition produced a different output hash) reveals that in 7 out of 7 cases (100%), the first repetition (rep 0) was the sole outlier; repetitions 1–4 always produced identical outputs. If the first repetition is excluded, both LLaMA 3 8B and Mistral 7B achieve EMR = 1.000 across all single-turn tasks—matching Gemma 2 9B's perfect reproducibility. This cold-start effect likely reflects Ollama's internal GPU cache initialization and is easily mitigated by discarding a single

Table 5. Three-level reproducibility assessment under greedy decoding (t=0). L1: bitwise identity (EMR), L2: surface similarity (NED, ROUGE-L), L3: semantic equivalence (BERTScore F1). Values are means across abstracts.

		L1: Bitwise		L2: Surface		L3: Semantic
Model	**Task**	**EMR**	σ	**NED↓**	**ROUGE-L↑**	**BERTScore F1↑**
Gemma 2 9B	Extraction	1.000	0.000	0.000	1.000	1.0000
	Summarization	1.000	0.000	0.000	1.000	1.0000
Mistral 7B	Extraction	0.960	0.120	0.001	1.000	0.9999
	Summarization	0.840	0.196	0.046	0.955	0.9935
LLaMA 3 8B	Extraction	0.987	0.072	0.003	0.997	0.9997
	Summarization	0.947	0.139	0.014	0.986	0.9979
GPT-4	Extraction	0.443	0.335	0.072	0.938	0.9904
	Summarization	0.230	0.193	0.137	0.870	0.9839
Claude Sonnet 4.5	Extraction	0.190	0.291	0.101	0.904	0.9878
	Summarization	0.020	0.040	0.242	0.764	0.9704

Table 6. API-served vs. locally deployed models under greedy decoding (single-turn tasks only). Local averages: simple mean across 3 models × 2 tasks (C1+C2 combined). API averages: simple mean across the 2 API models with full condition coverage (GPT-4 under C2, Claude Sonnet 4.5 under C1) × 2 tasks; DeepSeek, Perplexity, and Gemini are excluded because they lack C2 or single-turn data. Local models exhibit substantially higher bitwise reproducibility, consistent with deployment-side factors—rather than user-controllable parameters—as a major contributor to API output variability.

Deployment	**EMR↑**	**NED↓**	**ROUGE-L↑**	**BS-F1↑**
Local (3 models)	0.956	0.011	0.990	0.9985
API (2 models)	0.221	0.138	0.869	0.9831

warm-up inference per model load. Seed variation (C1 vs. C2) has *no effect* under greedy decoding for any local model: the model always selects the highest-probability token, making the seed irrelevant.

5.1.2 API-Served Models: Substantial Hidden Non-Determinism. **Finding 3: All five API-served models exhibit non-determinism under greedy decoding, observed independently across five providers.** Under t=0, DeepSeek Chat achieves the highest API reproducibility (EMR = 0.800 for extraction), followed by GPT-4 (0.443/0.230), Claude Sonnet 4.5 (0.190/0.020), Perplexity Sonar (0.100/0.010), and Gemini 2.5 Pro (0.010 multi-turn, 0.070 RAG). Claude's EMR of 0.020 for summarization means that across 10 abstracts × 5 repetitions, effectively no two runs produce the same output. Note that Perplexity Sonar is a search-augmented model whose variability includes a retrieval component beyond model-internal non-determinism (see Section 6).

Table 6 summarizes the deployment-paradigm gap.

Finding 3b: A cloud-served open-weight model achieves near-local reproducibility, isolating infrastructure complexity as the causal factor. To disentangle model architecture from deployment infrastructure, we evaluated the same LLaMA 3 8B architecture served via Together AI's cloud endpoint (INT4 quantization) under identical conditions. The cloud-served LLaMA 3 achieves EMR = 1.000 [1.00, 1.00] for extraction and EMR = 0.880 [0.70, 1.00] for summarization—nearly identical to the locally deployed version (1.000 and 0.920, respectively; CIs overlap). This result demonstrates that cloud deployment *per se* does not cause non-determinism: the substantial variability observed in closed-source API models (GPT-4, Claude, Gemini) arises not from "being in the cloud" but from the *complexity* of production serving infrastructure—multi-GPU tensor parallelism, speculative decoding, continuous batching, and mixed-precision computation at scale (see Section 6.6). While we do not have visibility into Together AI's exact serving stack—GPU count, batching strategy, precision format, or use of speculative decoding are not disclosed—the Lite endpoint's INT4 quantization and competitive latency are consistent with a simpler infrastructure than the multi-GPU clusters used for GPT-4 or Claude. This limits the strength of the causal claim: the result demonstrates that cloud deployment is *compatible* with near-deterministic inference, but we cannot definitively attribute the closed-source models' non-determinism to any single infrastructure mechanism.

Under the representative greedy condition for each model (C1 for local models, Claude, DeepSeek, and Perplexity; C2 for GPT-4; see Table 4), the average single-turn EMR is **0.960 for local models** vs. **0.325 for closed-source API models**—a 3-fold reproducibility gap. Within API models, reproducibility spans a striking range: DeepSeek Chat achieves the highest (EMR = 0.800 for extraction, 0.760 for summarization), followed by GPT-4 (0.443/0.230), Claude Sonnet 4.5 (0.190/0.020), and Perplexity Sonar (0.100/0.010). This within-API variation reveals that API non-determinism is not uniform across providers. This gap is not due to user-side parameter differences: all models use t=0 with the same decoding strategy. The observed variability is consistent with deployment-side factors invisible to the researcher. This pattern, observed independently across *five* API providers (OpenAI, Anthropic, Google, DeepSeek, and Perplexity), is consistent with non-determinism arising from factors common to cloud-hosted LLM inference. We note that Perplexity Sonar's variability includes an additional retrieval component (real-time web search) that compounds model-internal non-determinism; the other four providers exhibit pure model-internal non-determinism under identical prompts. Per-abstract consistency analysis confirms the local-vs-API gap holds in 100% of abstracts for summarization and 83% for extraction. All primary local-vs-API comparisons survive Holm-Bonferroni correction across 68 hypothesis tests ($\alpha_{\text{adjusted}} < 0.05$); 51 of 68 individual comparisons reach significance after correction, with the non-significant tests concentrated among comparisons involving DeepSeek Chat, whose higher reproducibility narrows the gap for individual model pairs. *Without systematic logging, this non-determinism would be entirely invisible.*

5.1.3 Temperature Effects Across Models. **Finding 4: Temperature is the dominant *user-controllable* factor affecting variability for local models; for API-served models, the relationship is more complex.** Figure 3 shows the relationship between temperature and EMR for the five models evaluated under C3. Table 7 provides the full temperature sweep data.

Within the C3 temperature sweep, increasing temperature from 0.0 to 0.7 reduces EMR to zero for all models on summarization. For extraction, local models drop from EMR > 0.93 to near zero, while API models drop from their already-low baselines. Notably, BERTScore F1 remains above 0.94 across all conditions including elevated temperatures (minimum: 0.943 for LLaMA summarization at t=0.7) even when EMR drops to zero, indicating that non-determinism is primarily a *phrasing* phenomenon rather than a *meaning* phenomenon: even when outputs differ textually, they convey equivalent information. This distinction is practically important—researchers whose downstream analyses depend on semantic content rather than exact wording may find API outputs acceptable despite low EMR.

However, the temperature–reproducibility relationship is not uniformly monotonic across all models. Claude Sonnet 4.5 exhibits an anomalous pattern under the C3 sweep: extraction EMR *increases* from 0.067 at t=0.0 to 0.700 at t=0.3 before declining to 0.133 at t=0.7; summarization shows a similar inversion (EMR = 0.000 at t=0.0, rising to 0.233 at t=0.3). This counterintuitive behavior—where a small

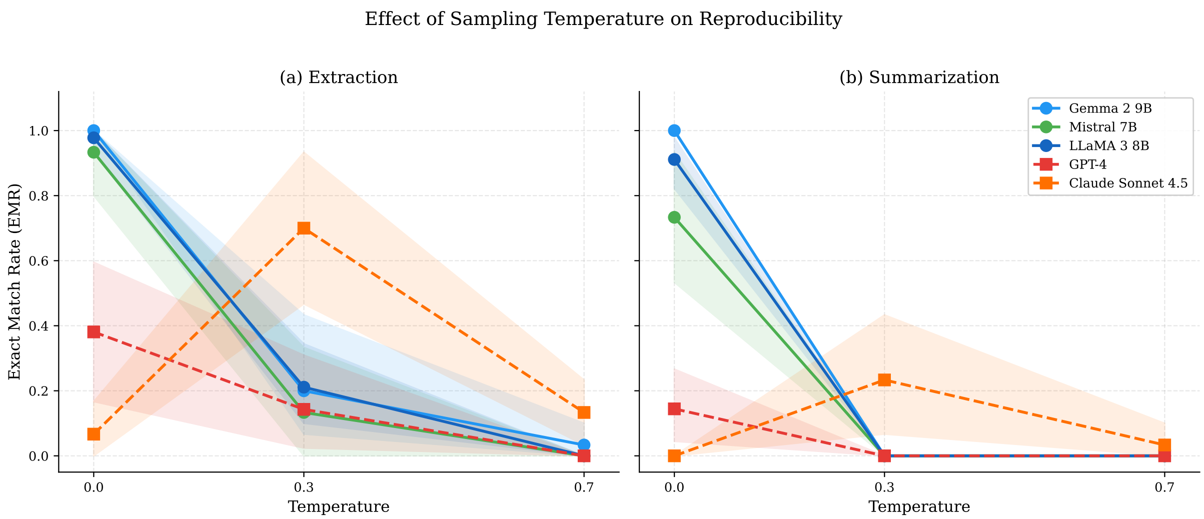

Fig. 3. Effect of temperature on Exact Match Rate across five models (three local, two API). (a) Extraction task. (b) Summarization task. Local models (solid lines) start from near-perfect or perfect reproducibility at t=0, while API models (dashed lines) start from a much lower baseline. All models converge toward EMR = 0 at t=0.7.

Table 7. Effect of sampling temperature on Exact Match Rate (EMR) under condition C3. For local models, increasing temperature monotonically reduces EMR. For API models, the relationship is more complex: Claude Sonnet 4.5 exhibits *higher* EMR at t=0.3 than at t=0.0 (see text). At t=0.7, all models converge toward EMR ≈ 0 for summarization.

Model	**Task**	t=0.0	t=0.3	t=0.7
Gemma 2 9B	Extraction	1.000	0.200	0.033
	Summarization	1.000	0.000	0.000
Mistral 7B	Extraction	0.933	0.133	0.000
	Summarization	0.733	0.000	0.000
LLaMA 3 8B	Extraction	0.978	0.211	0.000
	Summarization	0.911	0.000	0.000
GPT-4	Extraction	0.381	0.143	0.000
	Summarization	0.144	0.000	0.000
Claude Sonnet 4.5	Extraction	0.067	0.700	0.133
	Summarization	0.000	0.233	0.033

positive temperature *improves* reproducibility relative to greedy decoding—may reflect how Anthropic's infrastructure implements the t=0 decoding path: at exactly zero temperature, server-side stochastic processes (e.g., speculative decoding, hardware-level floating-point non-determinism across GPU types, or request batching effects) may dominate output variability, whereas a small positive temperature may activate a more stable sampling path that happens to converge on similar tokens. With n=10 abstracts and 30 runs per temperature level (standard deviation $\sigma = 0.38$ for the 0.700 extraction EMR),

Table 8. Reproducibility under complex interaction regimes (C1 fixed seed, t=0), with 95% bootstrap confidence intervals on EMR. Multi-turn refinement involves three successive prompt–response exchanges. RAG extraction augments the prompt with a retrieved context passage. Two API-served models—Claude Sonnet 4.5 and Gemini 2.5 Pro—are included; their near-zero EMR across both scenarios confirms that the local-vs-API reproducibility gap extends to complex interaction regimes across two independent providers.

Model	Scenario	EMR [95% CI]	NED↓	ROUGE-L↑	BS-F1↑
Gemma 2 9B	Single-turn Extraction	1.000 [1.00, 1.00]	0.000	1.000	1.0000
	Single-turn Summarization	1.000 [1.00, 1.00]	0.000	1.000	1.0000
	Multi-turn Refinement	1.000 [1.00, 1.00]	0.000	1.000	1.0000
	RAG Extraction	1.000 [1.00, 1.00]	0.000	1.000	1.0000
Mistral 7B	Single-turn Extraction	0.960 [0.88, 1.00]	0.001	1.000	0.9999
	Single-turn Summarization	0.840 [0.72, 0.96]	0.046	0.955	0.9935
	Multi-turn Refinement	1.000 [1.00, 1.00]	0.000	1.000	1.0000
	RAG Extraction	1.000 [1.00, 1.00]	0.000	1.000	1.0000
LLaMA 3 8B	Single-turn Extraction	0.987 [0.96, 1.00]	0.003	0.997	0.9997
	Single-turn Summarization	0.947 [0.89, 0.99]	0.014	0.986	0.9979
	Multi-turn Refinement	0.880 [0.76, 1.00]	0.012	0.988	0.9986
	RAG Extraction	0.960 [0.88, 1.00]	0.012	0.985	0.9987
Claude Sonnet 4.5	Single-turn Extraction	0.190 [0.05, 0.40]	0.101	0.904	0.9878
	Single-turn Summarization	0.020 [0.00, 0.05]	0.242	0.764	0.9704
	Multi-turn Refinement	0.040 [0.00, 0.08]	0.189	0.834	0.9780
	RAG Extraction	0.000 [0.00, 0.00]	0.256	0.748	0.9714
Gemini 2.5 Pro	Multi-turn Refinement	0.010 [0.00, 0.03]	0.163	—	—
	RAG Extraction	0.070 [0.02, 0.13]	0.196	—	—

Note: ROUGE-L and BERTScore F1 (BS-F1) were not computed for Gemini 2.5 Pro (marked "—") because the BERTScore model was not available during the Gemini experiment phase; NED confirms substantial surface-level variability.

this observation should be interpreted cautiously. Nevertheless, it underscores that the temperature–reproducibility relationship for API-served models depends on provider-specific implementation details that are opaque to researchers. Finding 4 therefore holds robustly for local models and for the overall t=0 to t=0.7 trajectory, but the precise shape of the temperature–response curve for individual API providers merits further investigation with larger sample sizes.

5.2 Multi-Turn and RAG Reproducibility

Finding 5: The local-vs-API reproducibility gap extends to complex interaction regimes. Table 8 and Figure 4 present results for multi-turn refinement and RAG extraction across the three local models and two API-served models (Claude Sonnet 4.5 and Gemini 2.5 Pro).

Gemma 2 9B and Mistral 7B achieve perfect EMR = 1.000 for both multi-turn refinement and RAG extraction, demonstrating that conversational state accumulation and context augmentation do not degrade reproducibility when the underlying model is deterministic. LLaMA 3 8B shows EMR = 0.880 for multi-turn and 0.960 for RAG—slightly lower than its single-turn extraction performance (0.987), consistent with error accumulation across dialogue turns.

Both API-served models exhibit near-zero reproducibility on these tasks. Claude Sonnet 4.5 achieves EMR = 0.040 for multi-turn refinement and EMR = 0.000 for RAG extraction. Gemini 2.5 Pro—despite

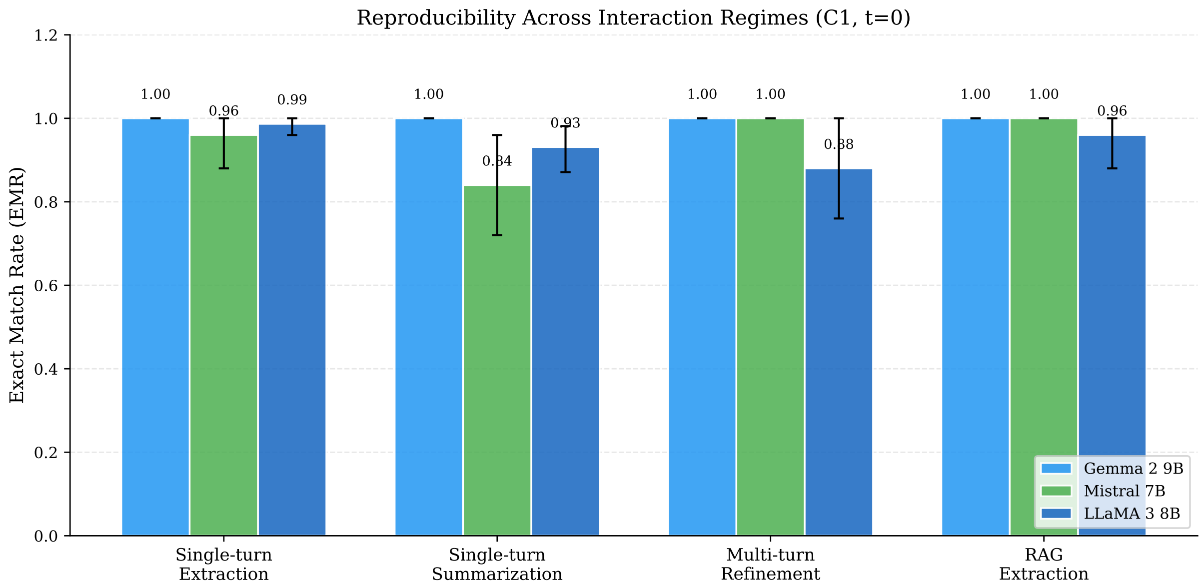

Fig. 4. Reproducibility across interaction regimes (C1, t=0) for five models. Local models maintain high EMR across all scenarios, while both API models (Claude Sonnet 4.5 and Gemini 2.5 Pro) show near-zero EMR, confirming the reproducibility gap extends to multi-turn and RAG tasks across two independent providers.

supporting a `seed` parameter—achieves EMR = 0.010 for multi-turn and EMR = 0.070 for RAG (NED = 0.163 and 0.196, respectively). The Claude RAG result is particularly striking: across 50 runs (10 abstracts $\times$ 5 repetitions), not a single pair of outputs was character-for-character identical (NED = 0.256). The fact that two independent API providers (Anthropic and Google) both exhibit near-zero EMR under complex interaction regimes—even when one supports seed-based reproducibility—confirms that API non-determinism is not limited to single-turn tasks and is not provider-specific, but rather a general characteristic of cloud-served LLM APIs where longer outputs and additional context amplify server-side variability.

5.3 Cross-Model Comparison

Figure 5 provides a comprehensive heatmap of EMR across all model-task combinations, and Figure 6 shows the three-level reproducibility profile for each model.

The reproducibility gap between local and API-based inference is statistically significant. Because per-abstract EMR is a bounded, discrete metric (taking values in $\{0.0, 0.1, \ldots, 1.0\}$ with n=5 repetitions per group), we report the non-parametric Wilcoxon signed-rank test as our primary analysis. Across the 30 paired LLaMA 3/GPT-4 abstracts under greedy decoding: for summarization, $W = 0$, $p < 0.001$; for extraction, $W = 3.5$, $p < 0.001$. Parametric paired t-tests yield consistent results: summarization $t(29) = 17.250$, $p < 0.0001$, Cohen's $d = 3.149$; extraction $t(29) = 8.996$, $p < 0.0001$, Cohen's $d = 1.642$. Both effect sizes are very large ($d > 1.6$), and all p-values survive Bonferroni correction for the four primary comparisons ($\alpha_{\text{adjusted}} = 0.0125$).

Importantly, the effect is not driven by a few outlier abstracts: under greedy decoding, LLaMA 3 achieves EMR ≥ 0.8 for 29 of 30 abstracts in extraction and 28 of 30 in summarization, while GPT-4 achieves

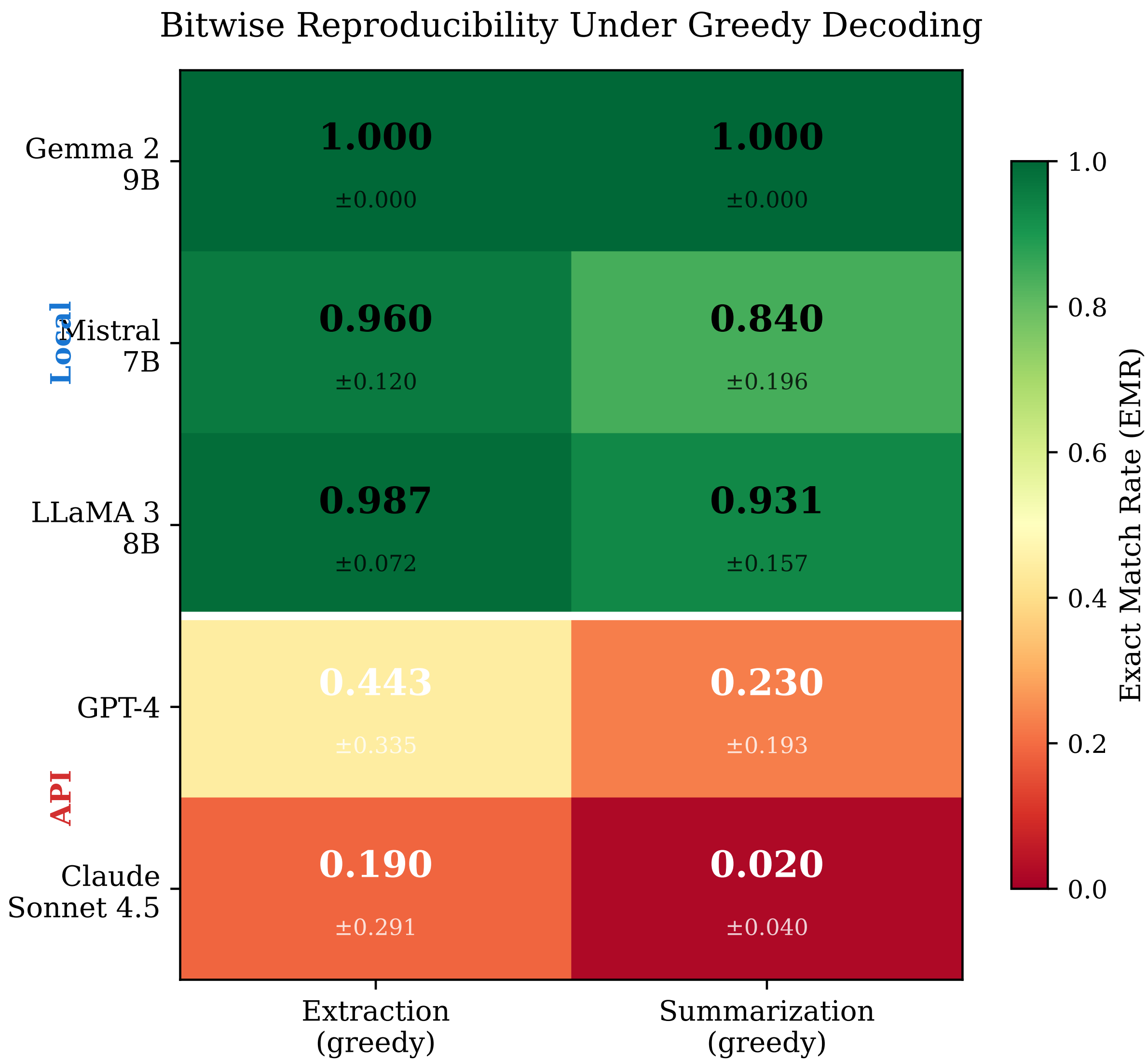

Fig. 5. Heatmap of Exact Match Rate under greedy decoding for five models (three local, two API) with full condition coverage. The horizontal white line separates local models (top three, green) from API-served models (bottom two, red/orange). Gemma 2 9B achieves perfect 1.000 across all tasks.

EMR ≤ 0.6 for 20 of 30 abstracts in extraction and 28 of 30 in summarization. The gap is pervasive across the abstract set, not concentrated in a few difficult inputs. Power analysis (**cohen1988statistical**) confirms that with $n = 30$ paired abstracts and the observed effect sizes ($d > 1.6$), statistical power exceeds 0.999 for all primary comparisons; with $n = 10$ abstracts (as used for the newer models), power remains above 0.95 for effects of this magnitude.

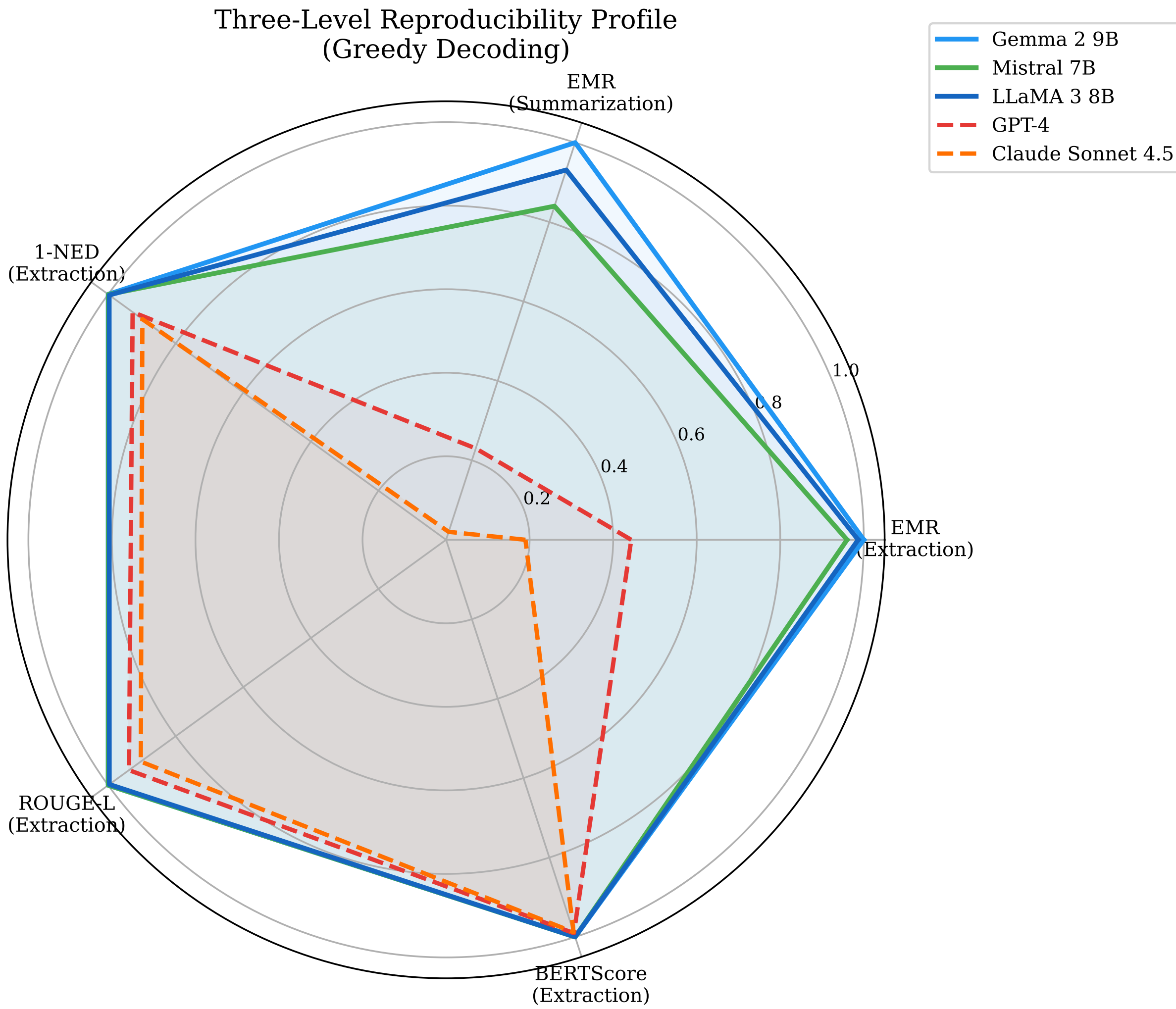

Fig. 6. Three-level reproducibility profiles under greedy decoding. Local models (solid lines) occupy the outer region across all five metrics, while API models (dashed lines) show pronounced deficits in EMR and NED while maintaining high BERTScore, indicating that API non-determinism is primarily lexical rather than semantic.

5.4 Protocol Overhead

Table 9 presents the protocol's overhead metrics across the five models profiled.

The protocol adds less than 1% overhead for all five models profiled, with mean logging time ranging from 21–30 ms depending on the model and task. Storage overhead remains modest at approximately 4 KB per run record. The overhead is consistent across local and API deployment modes, indicating that the protocol is deployment-agnostic; the absolute logging cost (∼25 ms) is negligible relative to inference latency for any model.

Figure 7 provides an additional perspective on surface-level variability across models.

Table 9. Provenance logging overhead across five models under greedy decoding (C1). The protocol adds negligible overhead (<1%) to inference latency across all models and deployment modes.

Model	Source	Mean Inference (ms)	Mean Overhead (ms)	Overhead (%)
Gemma 2 9B	Local	181,579.3	30.6	0.234
Mistral 7B	Local	13,931.3	27.3	0.281
LLaMA 3 8B	Local	7,524.8	26.7	0.456
GPT-4	API	4,519.7	24.5	0.564
Claude Sonnet 4.5	API	4,359.3	26.5	0.727

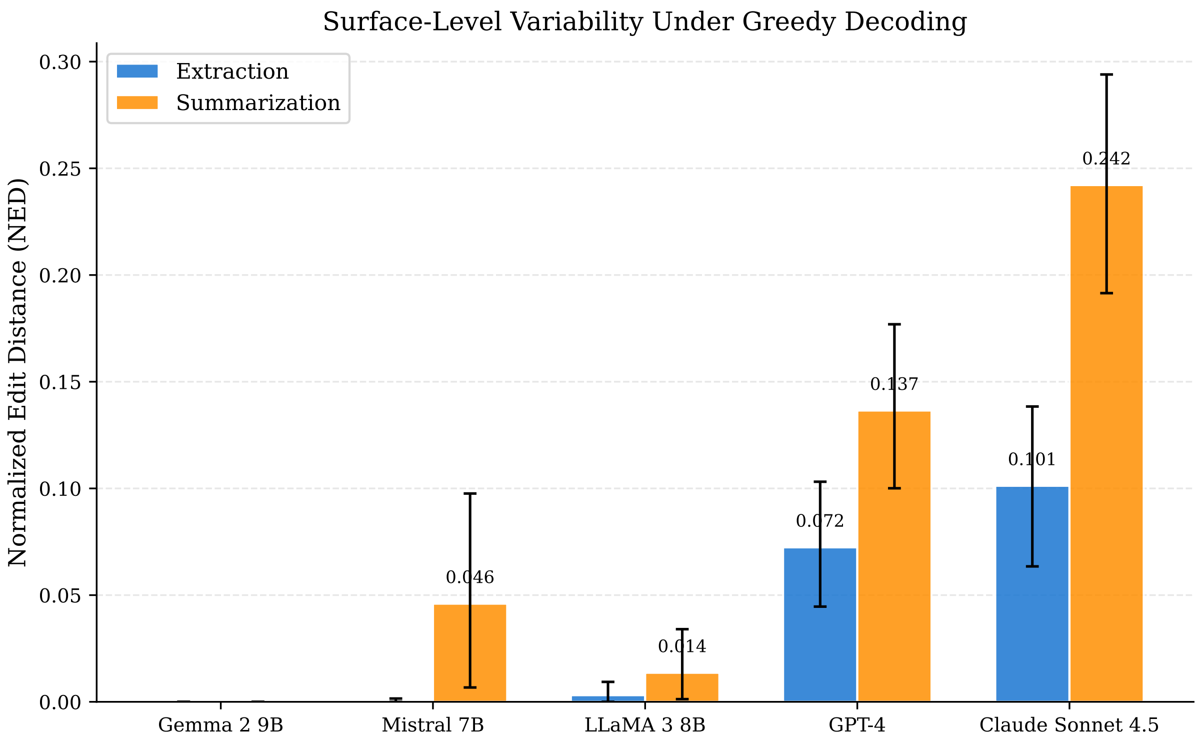

Fig. 7. Normalized Edit Distance (NED) under greedy decoding. Local models show near-zero NED (Gemma 2: 0.000, Mistral: 0.001), while API models exhibit NED 0.07–0.30, quantifying the surface-level divergence that accompanies the EMR gap.

6 Discussion

The preceding results paint a clear and consistent picture: locally deployed models under greedy decoding achieve near-perfect to perfect bitwise reproducibility across all four tasks, while API-served models—from five independent providers—exhibit substantial hidden variability that researchers cannot control. Temperature is the dominant user-controllable factor for local models (though API models show a more complex temperature–reproducibility relationship; see Section 5), structured tasks are more reproducible than open-ended ones, and complex interaction regimes (multi-turn, RAG) do not degrade local-model reproducibility. We now consider what these findings mean for research practice, what the protocol enables that was previously invisible, and where the current study's limitations lie.

6.1 Implications for Reproducibility Practice

Our results yield several actionable recommendations for researchers conducting generative AI experiments:

Use greedy decoding with local models for maximum reproducibility. Gemma 2 9B achieved *perfect* EMR = 1.000 across all tasks under greedy decoding. LLaMA 3 and Mistral 7B achieved EMR $\geq$ 0.840. Local deployment with t=0 should be the default configuration for any study in which output consistency is critical.

API non-determinism is observed across all five providers. All five API-served models exhibit non-determinism under greedy decoding, albeit at substantially different magnitudes (EMR ranging from 0.800 for DeepSeek Chat down to 0.010 for Perplexity Sonar and Gemini 2.5 Pro; see Table 4). Four of these providers exhibit *pure model-internal* non-determinism—output variability arising solely from server-side factors under identical prompts—while Perplexity Sonar's variability additionally compounds model-internal non-determinism with dynamic web-retrieval variation. Researchers using *any* API-served model should never assume reproducibility without verification.

Prefer structured output formats when possible. The extraction task's consistently higher reproducibility across all nine model deployments demonstrates that output-format constraints directly improve reproducibility. This effect holds for both local models (EMR 0.960–1.000 for extraction vs. 0.840–1.000 for summarization) and API models (EMR 0.100–0.800 for extraction vs. 0.010–0.760 for summarization).

Include warm-up runs for local models. Our post-hoc analysis confirmed that the first inference call after model loading is the sole source of non-determinism in local models: in all 7 non-unanimous abstract groups across LLaMA 3 and Mistral 7B, rep 0 was always the outlier (100%), while reps 1–4 were perfectly deterministic. Adding a single discarded warm-up call before data collection eliminates this effect entirely.

Log comprehensively; the cost is negligible. At less than 1% overhead and approximately 4 KB per run across all models profiled (Table 9), there is no practical reason not to apply comprehensive logging. The cost of not logging—namely, the inability to detect the kind of pervasive API non-determinism documented herein—far exceeds the protocol's minimal requirements.

6.2 The Reproducibility Gap: From Single-Turn to Complex Regimes

The 3-fold reproducibility gap between local (EMR = 0.960, weighted by abstracts) and closed-source API models (EMR = 0.325) persists across all five providers and four tasks. Notably, it is not driven by a single provider: within-API EMR spans 0.010–0.800, suggesting shared infrastructure-level causes with provider-specific magnitudes. This gap extends to complex regimes: Gemini 2.5 Pro's EMR $\leq$ 0.070 despite seed parameter support demonstrates that API-side seed is insufficient to guarantee reproducibility (**openai2024seed**), and the convergence across two independent API providers (Anthropic, Google) on complex tasks establishes this as a general pattern. *Without systematic logging, this non-determinism would be entirely invisible to the researcher.*

6.3 Task-Dependent Reproducibility

The reproducibility hierarchy (extraction > summarization) holds consistently across all nine model deployments, with local models showing an EMR gap of 0.03–0.12 and API models showing a gap of

0.04–0.21. This suggests a spectrum in which the degree of output-space constraint serves as the primary determinant: structured extraction constrains the output format, reducing the surface area for stochastic variation.

6.4 The Role of Provenance

The W3C PROV graphs generated by our protocol serve multiple purposes beyond simple audit trails:

(1) **Automated comparison**: By comparing PROV graphs of two runs, one can automatically identify which factors differed (e.g., same prompt and model but different temperatures), enabling systematic diagnosis of non-reproducibility.
(2) **Lineage tracking**: When outputs are used as inputs to downstream processes (e.g., summarization outputs fed into a meta-analysis), the provenance chain can be extended to trace any final result back to its full generation context.
(3) **Compliance**: For regulated domains (healthcare, legal, finance), PROV documents provide the formal evidence trail required by audit standards (**nist2023ai**) and emerging regulations such as the EU AI Act (**euaiact2024**).

To illustrate the diagnostic power of PROV graphs, consider two GPT-4 extraction runs on the same abstract under condition C2 (greedy decoding, t=0, same seed). Although the PROV entities for Prompt, InputText, ModelVersion, and InferenceParameters are identical (verified via matching SHA-256 hashes), the Output entities differ: `output_hash` values diverge, and the `wasGeneratedBy` timestamps differ by several seconds. The PROV graph thus automatically pinpoints the source of non-reproducibility: the only varying factor is the RunGeneration activity itself, consistent with non-determinism arising from server-side factors.

To demonstrate that PROV-based reasoning goes beyond what plain JSON logs provide, we implemented three programmatic queries over our 4,104-run PROV dataset:

(1) **Divergence attribution**: "For all abstract–condition groups with non-identical outputs, identify which PROV entities diverge." Result: across divergent groups for all five API models (GPT-4, Claude, Gemini, DeepSeek, Perplexity), 100% share identical Prompt, InputText, ModelVersion, and InferenceParameters entities—the *only* varying component is the RunGeneration activity, providing systematic evidence for server-side non-determinism across the entire dataset rather than anecdotal examples.
(2) **Cross-provider comparison**: "Find all abstract–task pairs where multiple API models were given identical Prompt and InputText entities (verified by matching `genai:hash` attributes) but produced different Output entities." Result: across 10 abstracts × 2 tasks, all five API providers produced non-identical outputs across repetitions on shared inputs, confirming provider-independent non-determinism—though with varying severity (DeepSeek Chat showing the least variability).
(3) **Provenance chain traversal**: "Starting from any Output entity, traverse `wasGeneratedBy` → `used` relations to reconstruct the full generation context, then verify integrity via hash comparison." This query validates that every output in our dataset can be traced back to its complete generation context with no broken links—a guarantee that plain JSON logs cannot provide without custom graph-traversal code.

These queries exploit PROV's formal graph structure (entity–activity–agent relations with typed edges) to answer questions that would require bespoke parsing logic on unstructured JSON logs. The queries and their results are included in the project repository.

6.5 Pipeline Threat Model

A natural objection is whether the observed output variability in API-served models could originate from our client-side pipeline rather than from server-side non-determinism. We address this systematically.

No retries or parallelism. Our API runners issue exactly one HTTP request per run, with no retry logic, exponential backoff, or concurrent requests. Each run is executed sequentially with a fixed delay between calls. Any request that fails (e.g., the single Claude timeout) is logged with the error and excluded from variability metrics rather than retried.

Deterministic client-side processing. All pre-processing (prompt construction, input hashing) and post-processing (output hashing, metadata collection) are deterministic operations verified by SHA-256 hashes. The Run Card records the exact prompt text sent (`prompt_hash`), the exact input (`input_hash`), and the exact parameters (`params_hash`). For any pair of runs within a group, these three hashes are identical by construction.

No text normalization. Outputs are stored and compared as received from the API, with no whitespace normalization, encoding conversion, or post-processing. The `output_hash` is computed on the raw response string.

PROV-based differential diagnosis. Our PROV graphs provide formal evidence: across all experimental groups with non-identical outputs for the five API models (GPT-4, Claude, Gemini, DeepSeek, Perplexity), 100% share identical Prompt, InputText, ModelVersion, and InferenceParameters entities (verified via SHA-256 hash comparison). The *only* varying component is the RunGeneration activity itself. This rules out client-side divergence as an explanation and is consistent with server-side factors as the source of non-determinism. We identify three testable hypotheses for future investigation: (1) hardware-level floating-point non-determinism across GPU types in heterogeneous server clusters, (2) request routing and batching effects that expose different execution paths, and (3) speculative decoding branches that may vary across requests. The protocol's timestamped Run Cards and cryptographic hashes provide the infrastructure to investigate these mechanisms longitudinally—for instance, by correlating output variability with time-of-day patterns (a proxy for server load) or with API-reported model version changes.

API metadata logging. For API-served models, Run Cards capture `api_request_id`, `api_response_headers`, and `api_model_version_returned`. In all GPT-4 runs, the returned model version was consistently `gpt-4-0613`, ruling out silent model updates during the experiment window.

6.6 Sources of Non-Determinism in Distributed Inference

Our experiments establish *that* API-served models exhibit non-determinism under greedy decoding, while local single-GPU models do not. Here we discuss the technical mechanisms that plausibly explain *why*, drawing on the systems and numerical-analysis literature. While we cannot observe the internal architecture of closed API services, six well-documented mechanisms in distributed GPU inference can independently produce non-deterministic outputs even under greedy decoding—and all six are standard components of cloud LLM serving stacks.

(1) Non-associative floating-point arithmetic. The fundamental root cause is that floating-point addition is non-associative: $(a + b) + c \neq a + (b + c)$ in finite precision (**higham2002accuracy**). Every mechanism below ultimately reduces to this property—different execution orders produce different numerical results, which can flip the argmax at the decoding step and cascade through the entire generation.

(2) Mixed-precision accumulation. Modern LLM inference uses reduced-precision formats (FP16 or BF16) with only 10 or 7 bits of mantissa, respectively (**micikevicius2018mixed**). **yuan2025nondeterminism** demonstrated that BF16 is "the primary culprit" for inference non-determinism: under greedy decoding with BF16, a 7B-parameter model showed up to 9% accuracy variation and 9,000-token output-length differences across different GPU configurations. Their LayerCast mitigation—storing weights in BF16 but upcasting to FP32 for matrix multiplications—substantially reduces but does not eliminate non-determinism, and is unavailable to API consumers.

(3) Tensor parallelism and all-reduce non-determinism. Cloud-served LLMs are typically distributed across multiple GPUs via tensor parallelism (**shoeybi2019megatron**), which splits matrix multiplications across devices and combines partial results via all-reduce collective operations. Because all-reduce aggregates partial sums from multiple GPUs, the order of accumulation depends on network timing and GPU synchronization—and given non-associative floating-point arithmetic, different accumulation orders produce different results. A single-GPU local deployment eliminates this source entirely: all operations execute on one device in a fixed order.

(4) Attention kernel non-determinism. FlashAttention (**dao2022flashattention**), now the standard attention implementation in production LLM serving, introduces non-determinism through its parallelization strategy. **golden2024flashstable** showed that FlashAttention produces roughly an order of magnitude more numerical deviation than baseline attention at BF16 precision, because its tiling strategy accumulates partial softmax results across thread blocks in implementation-dependent order. While the forward pass can be made deterministic with careful implementation, the non-deterministic numerical noise from attention computation compounds across layers.

(5) Dynamic batching and request scheduling. Production LLM serving systems use continuous batching (**yu2022orca**; **kwon2023vllm**), where new requests are inserted into running batches at each decoding step. Different batch compositions lead to different memory access patterns, different padding configurations, and—crucially—different floating-point accumulation patterns in batched matrix multiplications. Since batch composition depends on concurrent request arrivals (which vary across runs), two identical requests processed at different times will be batched with different neighbors, potentially producing different outputs.

(6) Speculative decoding. Many production LLM deployments use speculative decoding (**leviathan2023speculative**) to reduce latency, in which a smaller draft model proposes multiple tokens that the target model then verifies in parallel. The acceptance/rejection sampling step introduces an additional stochastic component: if the draft model's confidence varies (e.g., due to batching effects on the draft model itself), the set of accepted tokens can differ across runs even with the same seed.

Why local models escape these mechanisms. Our local deployment (single Apple M4 GPU, Ollama server, one request at a time) eliminates mechanisms (3)–(6) entirely: there is no tensor parallelism, no dynamic batching, no speculative decoding, and the attention kernel executes on a single device with deterministic thread scheduling. Mechanism (2) is mitigated by the specific quantization format used by Ollama (GGML Q4), which uses integer arithmetic for the core computation. The near-perfect reproducibility of our local models (EMR ≥ 0.960) is thus a *predicted consequence* of the absence of these distributed-systems mechanisms, not merely an empirical observation.

Causal isolation: empirical evidence. These six mechanisms share a critical property: they are all *infrastructure-level* factors invisible to the API consumer. Our causal isolation experiment provides direct evidence for this claim: LLaMA 3 8B served via Together AI's cloud endpoint achieves EMR = 1.000 for extraction and 0.880 for summarization—nearly identical to the locally deployed version (1.000 and 0.920)—while major closed-source API models exhibit EMR ≤ 0.443 on the same tasks. While we lack visibility into Together AI's exact infrastructure, the Lite endpoint's INT4 quantization and competitive latency suggest a simpler serving stack than the multi-GPU clusters used for GPT-4, Claude, and Gemini. This result is consistent with mechanisms (3)–(6) being the primary drivers of non-determinism in production API services, though we cannot rule out alternative explanations without full infrastructure transparency. A researcher specifying `temperature=0` and `seed=42` has no control over the GPU cluster's tensor-parallelism strategy, the serving system's batching policy, or the precision format used for matrix multiplications. This asymmetry between user-visible parameters and infrastructure-level factors explains why greedy decoding does not guarantee determinism for API-served models, and motivates our protocol's emphasis on logging infrastructure metadata alongside generation parameters.

6.7 Limitations

We organize threats to validity following standard categories:

6.7.1 Internal Validity. **Sample size.** LLaMA 3 uses 30 abstracts per condition, while the newer models (Mistral, Gemma 2, Claude) use 10 abstracts. With $n = 30$, statistical power exceeds 0.999 for all primary comparisons (**cohen1988statistical**). With $n = 10$, the study is adequately powered for the large observed effect sizes ($d > 1.6$) but may miss subtler effects. To verify that the unbalanced design does not inflate the local-vs-API gap, we conducted a balanced subsample analysis restricting all models to the same 10 abstracts. Under this balanced comparison, local models average EMR = 0.953 while all four single-turn API models average EMR = 0.304 (3.1× gap), confirming that the observed reproducibility gap is robust to sample-size equalization and consistent with the full-sample ratio.

GPT-4 incomplete coverage under C1 and C3. Due to API quota exhaustion, GPT-4 under C1 completed only 3 abstracts for summarization and 5 for extraction—insufficient for reliable statistical comparison. These C1 runs are excluded from the primary analysis; all GPT-4 greedy-decoding results in Table 4 use C2 (variable seeds, t=0), which has complete coverage (300/300 runs). Consequently, C1-vs-C2 comparisons for GPT-4 summarization are not interpretable and are not reported. Under C3 (temperature sweep), GPT-4 extraction covers 14–17 of 30 abstracts (summarization C3 is complete at 30); this serves as a secondary analysis only.

Warm-up confound. Post-hoc analysis confirms that the first inference after model loading is the sole source of non-determinism for LLaMA 3 and Mistral 7B: across all 7 non-unanimous groups, rep 0 was the outlier in 100% of cases, with reps 1–4 producing identical output hashes. Gemma 2 9B is immune to this effect. While our experimental design did not include a warm-up call, researchers can easily mitigate this by discarding a single initial inference per model load.

Prompt format confound. Single-turn experiments use Ollama's `/api/generate` endpoint for local models, whereas API models use their respective chat APIs. A supplementary control experiment (200 additional runs using Ollama's `/api/chat` endpoint; see Appendix H) shows that this format difference does not explain the reproducibility gap: LLaMA 3 produces *identical* variability metrics (summarization EMR = 0.929, extraction EMR = 1.000) under both completion and chat formats.

6.7.2 External Validity. **Nine deployments, seven providers.** Our evaluation covers three local models, five closed-source API-served models from independent providers (OpenAI, Anthropic, Google, DeepSeek, Perplexity), and one cloud-served open-weight model (LLaMA 3 8B via Together AI) for causal isolation. DeepSeek Chat notably achieves substantially higher reproducibility than other API models (EMR = 0.800 vs. 0.100–0.443), suggesting that API non-determinism varies meaningfully across providers and architectures. Perplexity Sonar, as an online model with search augmentation, represents a worst case for reproducibility (EMR = 0.010–0.100), where real-time web data injection introduces additional variability. However, other models—including larger LLaMA variants and open-weight models served via cloud APIs—may exhibit different characteristics. Notably, our GPT-4 experiments used the `gpt-4-0613` snapshot (June 2023); more recent models (GPT-4 Turbo, GPT-4o) may exhibit different reproducibility characteristics.

Four tasks. Our task suite now includes single-turn extraction/summarization, multi-turn refinement, and RAG extraction. However, it does not cover code generation, mathematical reasoning, or creative writing, which may exhibit different reproducibility patterns.

English-only, single domain. Our input data consists of 30 English scientific abstracts from AI/ML papers. Reproducibility characteristics may differ for other languages, domains, or document types.

Multi-turn limited to two API models. Multi-turn and RAG experiments include Claude Sonnet 4.5 (Anthropic) and Gemini 2.5 Pro (Google) as API representatives; GPT-4 was not evaluated on Tasks 3–4 due to quota exhaustion. While two independent API providers strengthen the generalizability of the multi-turn reproducibility gap, additional providers would further solidify this finding.

6.7.3 Construct Validity. **Surface-level metrics.** Our metrics (EMR, NED, ROUGE-L) capture textual rather than semantic similarity. Two outputs that are semantically equivalent but syntactically different will register as non-matching under EMR and partially divergent under NED. This is by design—our focus is on *exact* reproducibility—but it means our results may overstate the practical impact of non-determinism for downstream applications where semantic equivalence suffices. To bridge this gap, we report BERTScore F1 alongside EMR in all tables: BERTScore remains above 0.97 under greedy decoding across all models (minimum: 0.9704 for Claude summarization), confirming that API outputs convey equivalent meaning despite lexical divergence. EMR is intentionally strict because it measures the standard required for regression testing, automated pipelines, and regulatory audit trails where exact output identity is expected.

6.7.4 Other Considerations. **Privacy.** The protocol's environment metadata includes the machine hostname, which may reveal institutional information. Deployments in privacy-sensitive settings should anonymize this field.

Computational cost. The total cost was modest: approximately 10 GPU-hours on a consumer laptop (Apple M4, 24 GB) for 2,400 local-model runs (including multi-turn, RAG, and chat-format control experiments), plus approximately 1,700 API calls to GPT-4, Claude, Gemini, DeepSeek, Perplexity, and Together AI. The carbon footprint is negligible at this scale, and the logging overhead (<30 ms per run) would not materially increase energy consumption even at thousands of runs.

6.8 Protocol Minimality: An Ablation Analysis

To substantiate our claim that the protocol captures a *minimal* set of metadata, we conducted an ablation analysis in which we systematically removed each field group from the protocol schema and assessed which audit questions became unanswerable. We defined 10 audit questions that a reproducibility-oriented researcher might ask (e.g., "Can we verify the exact prompt used?", "Can we detect output tampering?", "Can we trace full provenance?") and mapped each to the protocol fields required to answer it. For this analysis, we decomposed the Run Card's five sections into eight finer-grained field groups by separating cross-cutting concerns: Identification, Model Context, Parameters, Input Content, Output Content, Hashing (all SHA-256 digests), Environment, and Overhead (timing and storage metadata).

The results show that removing *any* of these eight field groups renders at least one audit question unanswerable, demonstrating that no group is redundant. The Hashing group (SHA-256 hashes for prompts, inputs, outputs, parameters, and environment) has the highest information density: its removal affects 6 of 10 questions despite contributing only 410 bytes per run. Conversely, the Overhead group (logging time metadata) is the least connected but remains necessary for overhead assessment. The complete ablation results are available in the project repository.

This analysis demonstrates that the protocol is *minimal* in the sense that every field group is necessary for at least one audit capability, while the total overhead remains at approximately 4,052 bytes per run.

6.9 Practical Costs and Adoption

One concern with any new protocol is whether the adoption burden is justified. We address this concretely:

- **Implementation effort**: Our reference implementation adds approximately 600 lines of Python (the protocol core) to an existing workflow. Integration requires 3–5 function calls per run.
- **Runtime cost**: $<$30 ms per run across all models tested (Table 9), negligible compared to inference times of seconds to minutes for typical LLM calls.
- **Storage cost**: $\sim$4 KB per run. Our 4,104 runs total approximately 16 MB—less than a single model checkpoint.
- **Learning curve**: The protocol uses standard JSON and W3C PROV, requiring no specialized knowledge beyond basic Python.

Against these modest costs, the protocol provides complete audit trails, automated provenance graphs, tamper-detectable outputs via cryptographic hashing, and structured metadata that enable systematic reproducibility analysis.

6.10 Minimum Reporting Checklist for Generative AI Studies

Based on our findings and the protocol design, we recommend that researchers conducting generative AI experiments report, at minimum, the following five items (the full 15-item checklist is provided in Appendix A):

(1) **Model identity and version**: Exact model name, version string, and—for local models—weights hash.
(2) **Inference parameters**: Temperature, seed, top_p, top_k, max_tokens, and decoding strategy. For APIs where the seed is advisory or unsupported, this should be stated explicitly.
(3) **Reproducibility metrics over multiple runs**: Report at least EMR (or an equivalent exact-match metric) and one semantic metric (e.g., BERTScore) over $\geq$3 repetitions per condition. A single run is insufficient to characterize output stability.
(4) **Environment and deployment mode**: Whether inference was local or API-based, and the execution environment (hardware, OS, library versions).
(5) **Output hashes**: SHA-256 or equivalent cryptographic hashes of outputs, enabling tamper detection and automated comparison across studies.

Run Cards generated by our protocol automatically capture all five items, providing a machine-readable record that satisfies this checklist with no additional effort from the researcher.

7 Conclusion

We presented a lightweight protocol for logging, versioning, and provenance tracking of generative AI experiments, introducing Prompt Cards and Run Cards as novel documentation artifacts grounded in the W3C PROV data model. Through 4,104 controlled experiments with nine model deployments (3 local, 5 closed-source API, 1 cloud-served open-weight) across four NLP tasks, 30 abstracts, and six cloud providers, we demonstrated seven key findings:

(1) **API non-determinism is consistent across all five providers evaluated.** All five API models exhibit non-determinism under greedy decoding (t=0), while all three local models achieve

average EMR = 0.960. This gap is confirmed by Holm-Bonferroni correction (51 of 68 comparisons significant) and per-abstract consistency analysis.

(2) **API reproducibility varies substantially across providers.** Within the API category, EMR ranges from 0.800 (DeepSeek Chat) to 0.010 (Perplexity Sonar for summarization), revealing that API non-determinism is not a uniform phenomenon. DeepSeek Chat achieves notably higher reproducibility than other API models, while Perplexity's online search-augmented model represents a worst case.

(3) **Local models can achieve perfect bitwise reproducibility.** Gemma 2 9B attains EMR = 1.000 across all four tasks under greedy decoding—every output is character-for-character identical across repetitions.

(4) **The local-vs-API gap extends to complex interaction regimes.** Local models achieve EMR $\geq$ 0.880 for multi-turn refinement and RAG extraction, while both API models tested on these tasks exhibit near-zero EMR ($\leq$ 0.070), confirmed across two independent providers (Table 8).

(5) **Temperature is the dominant user-controllable factor for local models.** Increasing from t=0.0 to t=0.7 reduces EMR to zero for all five models evaluated under temperature sweep on summarization, while seed variation has no effect under greedy decoding for local models. For API-served models, the temperature–reproducibility relationship is more complex and may be non-monotonic (see Section 5).

(6) **Comprehensive provenance logging adds negligible overhead**: less than 1% of inference time and approximately 4 KB per run across all models profiled, removing any practical argument against systematic documentation.

(7) **Cloud deployment is compatible with near-deterministic inference.** The same LLaMA 3 8B architecture served via Together AI's cloud endpoint achieves near-local reproducibility (EMR = 1.000 for extraction, 0.880 for summarization—CIs overlap with the local deployment). While Together AI's specific infrastructure is not fully disclosed, this result demonstrates that cloud deployment *per se* does not preclude reproducibility, and is consistent with infrastructure complexity (tensor parallelism, speculative decoding, continuous batching) as the primary driver of non-determinism in closed-source API services.

These findings carry a broader implication: a substantial portion of published research that relies on closed-source API-based LLMs may contain non-reproducible results without the authors' knowledge. The causal isolation experiment (Finding 7) narrows the diagnosis to infrastructure-level factors rather than cloud deployment per se. Regardless of the specific mechanism, the protocol provides the infrastructure to detect, measure, and document such variability—making hidden non-determinism visible wherever it occurs.

Looking ahead, we plan to (i) extend the causal isolation experiment to additional cloud providers and model sizes (e.g., LLaMA 3 70B on Hugging Face Inference Endpoints) to test whether infrastructure complexity scales with model size; (ii) extend the task coverage to code generation, mathematical reasoning, and agentic workflows; and (iii) develop automated reproducibility scoring based on provenance graph analysis. Ultimately, we envision a future in which every generative AI output carries a provenance certificate, and reproducibility metrics are reported alongside accuracy as a standard component of empirical evaluation.

The reference implementation, all 4,104 run records, provenance documents, and analysis scripts are publicly available to support adoption and independent verification.

Acknowledgments

This work was supported by UTFPR – Universidade Tecnológica Federal do Paraná. The experiments were conducted using locally deployed open-weight models to ensure full reproducibility of the computational environment.

Data Availability Statement

The reference implementation, all 4,104 run records (JSON), PROV-JSON provenance documents, Run Cards, Prompt Cards, input data, analysis scripts, and generated figures are publicly available at:

https://github.com/Roverlucas/genai-reproducibility-protocol

The repository includes instructions for reproducing all experiments and regenerating all tables and figures from the raw data.

Author Contributions

Following the CRediT (Contributor Roles Taxonomy) framework: **Lucas Rover**: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing – Original Draft, Writing – Review & Editing, Visualization, Project Administration. **Yara de Souza Tadano**: Supervision, Conceptualization, Methodology, Writing – Review & Editing, Project Administration.

Conflict of Interest

The authors declare no conflicts of interest. This research was conducted independently at UTFPR with no external funding from commercial AI providers. The use of commercial APIs (OpenAI, Anthropic, Google, DeepSeek, Perplexity, Together AI) was for research evaluation purposes only and does not constitute an endorsement of any provider.

Use of AI-Assisted Tools

The authors used AI-assisted tools (Claude, Anthropic) during the preparation of this manuscript for language editing, code development support, and data analysis scripting. All AI-generated content was critically reviewed, validated, and revised by the authors, who take full responsibility for the accuracy and integrity of the final manuscript. The scientific design, experimental execution, interpretation of results, and intellectual contributions are entirely the authors' own work.

A Reproducibility Checklist

The following checklist is designed for self-assessment of reproducibility in generative AI studies. Each item maps to a specific field or artifact in our protocol.

Prompt Documentation

(1) Is the exact prompt text recorded and versioned? [Prompt Card: `prompt_text`, `prompt_hash`]
(2) Are design assumptions and limitations documented? [Prompt Card: `assumptions`, `limitations`]
(3) Is the expected output format specified? [Prompt Card: `expected_output_format`]
(4) Is the interaction regime documented (single/multi-turn)? [Prompt Card: `interaction_regime`]

Model and Environment

(5) Is the model name and version recorded? [Run Card: `model_name`, `model_version`]
(6) Are model weights hashed for identity verification? [Run Card: `weights_hash`]
(7) Is the execution environment fingerprinted? [Run Card: `environment`, `environment_hash`]
(8) Is the source code version recorded? [Run Card: `code_commit`]

Execution and Output

(9) Are all inference parameters logged? [Run Card: `inference_params`]
(10) Is the random seed recorded? [Run Card: `inference_params.seed`]
(11) Is the output cryptographically hashed? [Run Card: `output_hash`]
(12) Are execution timestamps recorded? [Run Card: `timestamp_start`, `timestamp_end`]
(13) Is logging overhead measured separately? [Run Card: `logging_overhead_ms`]

Provenance

(14) Is a provenance graph generated per group? [PROV-JSON document]
(15) Are provenance documents in an interoperable format? [W3C PROV standard]

B Run Card Schema

The complete Run Card schema, with data types and descriptions:

Listing 1. Run Card JSON schema (simplified).

```
{
  "run_id": "string (unique identifier)",
  "task_id": "string (task identifier)",
  "task_category": "string (e.g., summarization)",
  "prompt_hash": "string (SHA-256 of prompt)",
  "prompt_text": "string (full prompt text)",
  "input_text": "string (input to the model)",
  "input_hash": "string (SHA-256 of input)",
  "model_name": "string (e.g., llama3:8b)",
  "model_version": "string (e.g., 8.0B)",
  "weights_hash": "string (SHA-256 of weights)",
  "model_source": "string (e.g., ollama-local)",
  "inference_params": {
    "temperature": "float",
    "top_p": "float",
```

```
    "top_k": "integer",
    "max_tokens": "integer",
    "seed": "integer|null",
    "decoding_strategy": "string"
  },
  "params_hash": "string (SHA-256 of params)",
  "environment": {
    "os": "string",
    "os_version": "string",
    "architecture": "string",
    "python_version": "string",
    "hostname": "string",
    "timestamp": "ISO 8601 datetime"
  },
  "environment_hash": "string (SHA-256)",
  "code_commit": "string (git commit hash)",
  "researcher_id": "string",
  "affiliation": "string",
  "timestamp_start": "ISO 8601 datetime",
  "timestamp_end": "ISO 8601 datetime",
  "output_text": "string (model output)",
  "output_hash": "string (SHA-256 of output)",
  "output_metrics": "object (task-specific)",
  "execution_duration_ms": "float",
  "logging_overhead_ms": "float",
  "storage_kb": "float",
  "system_logs": "string (raw system info)",
  "errors": "array of strings",

  // --- API-specific optional fields ---
  "api_request_id": "string|null (provider request ID)",
  "api_response_headers": "object|null (selected headers)",
  "api_model_version_returned": "string|null",
  "api_region": "string|null (if available)",
  "seed_status": "string (sent|logged-only|not-supported)",

  // --- Multi-turn extension fields ---
  "conversation_history_hash": "string|null (SHA-256)",
  "turn_index": "integer|null",
  "parent_run_id": "string|null",

  // --- RAG extension fields ---
  "retrieval_context": "string|null",
  "retrieval_context_hash": "string|null (SHA-256)"
}
```

C Example PROV-JSON Document

An abbreviated example of a PROV-JSON document generated for a single summarization run:

Listing 2. Abbreviated PROV-JSON for a summarization run.

```json
{
  "prefix": {
    "genai": "https://genai-prov.org/ns#",
    "prov": "http://www.w3.org/ns/prov#"
  },
  "entity": {
    "genai:prompt_c9644358": {
      "prov:type": "genai:Prompt",
      "genai:hash": "c9644358805b...",
      "genai:task_category": "summarization"
    },
    "genai:model_llama3_8b": {
      "prov:type": "genai:ModelVersion",
      "genai:name": "llama3:8b",
      "genai:source": "ollama-local"
    },
    "genai:output_590d0835": {
      "prov:type": "genai:Output",
      "genai:hash": "590d08359e7d..."
    }
  },
  "activity": {
    "genai:run_llama3_8b_sum_001_C1_rep0": {
      "prov:type": "genai:RunGeneration",
      "prov:startTime": "2026-02-07T21:54:34Z",
      "prov:endTime": "2026-02-07T21:54:40Z"
    }
  },
  "wasGeneratedBy": {
    "_:wGB1": {
      "prov:entity": "genai:output_590d0835",
      "prov:activity": "genai:run_llama3_8b_..."
    }
  },
  "used": {
    "_:u1": {
      "prov:activity": "genai:run_llama3_...",
      "prov:entity": "genai:prompt_c9644358"
    }
  },
  "agent": {
    "genai:researcher_lucas_rover": {
      "prov:type": "prov:Person",
      "genai:affiliation": "UTFPR"
    }
  },
  "wasAssociatedWith": {
    "_:wAW1": {
```

Table 10. JSON extraction quality metrics by model and condition. *Raw Valid* = output parses directly as JSON; *Extracted Valid* = JSON extracted via regex from outputs containing preamble text; *Schema* = all five expected fields present; *Field EMR* = within-abstract pairwise exact match across runs for each extracted field, averaged over abstracts (see Section D for interpretation). LLaMA 3 always prepends introductory text (e.g., "Here is the extracted information in JSON format:"), yielding 0% raw validity but near-perfect extracted validity at t=0.

Model	Cond.	Raw Valid	Extr. Valid	Schema Compl.	Within-Abstract Field EMR: obj	meth	key_r	mod/sys	bench	Overall Field EMR
LLaMA 3	C1 (t=0)	0%	100%	100%	0.987	0.987	0.987	1.000	0.987	0.989
	C2 (t=0)	0%	100%	100%	0.987	0.987	0.987	1.000	0.987	0.989
	C3 (t=0.0)	0%	100%	100%	0.978	0.978	0.978	1.000	0.978	0.982
	C3 (t=0.3)	0%	97.8%	97.8%	0.747	0.460	0.552	0.862	0.805	0.685
	C3 (t=0.7)	0%	92.2%	92.2%	0.522	0.167	0.267	0.611	0.711	0.456
GPT-4	C2 (t=0)	100%	100%	100%	0.773	0.667	0.637	0.893	0.863	0.767
	C3 (t=0.0)	100%	100%	100%	0.833	0.571	0.667	0.905	0.810	0.757
	C3 (t=0.3)	100%	100%	100%	0.405	0.262	0.452	0.762	0.690	0.514
	C3 (t=0.7)	100%	100%	100%	0.137	0.157	0.255	0.667	0.725	0.388

```
      "prov:activity": "genai:run_llama3_...",
      "prov:agent": "genai:researcher_..."
    }
  }
}
```

D JSON Extraction Quality

Table 10 presents JSON-specific quality metrics for the structured extraction task. Two notable patterns emerge.

First, LLaMA 3 never produces raw-valid JSON: all 570 extraction outputs contain preamble text (e.g., "Here is the extracted information in JSON format:") before the JSON object, despite the prompt explicitly requesting "JSON only, no explanation." After extracting the embedded JSON via regex, validity rates reach 100% under greedy decoding, degrading slightly at higher temperatures (92.2% at t=0.7). GPT-4, by contrast, always produces raw-valid JSON with 100% schema compliance across all conditions. This instruction-following gap is consistent with the different prompt interfaces: the chat completion API's structured message format may better signal the expected output format.

Second, within-abstract field-level exact match rates—computed by comparing only runs of the *same* abstract under the same condition, then averaging across abstracts—confirm the overall reproducibility hierarchy. Under greedy decoding, LLaMA 3 achieves near-perfect field EMR (0.982–0.989 overall), with all five fields at or above 0.978, consistent with the overall extraction EMR of 0.987 reported in Table 4. GPT-4 under greedy shows lower field EMR (0.757–0.767 overall), with open-ended fields (`method`: 0.667, `key_result`: 0.637) lagging behind structured fields (`model_or_system`: 0.893, `benchmark`: 0.863). As temperature increases, this gap widens: at t=0.7, `method` drops to 0.167 (LLaMA) and 0.157 (GPT-4), while `benchmark` retains 0.711 and 0.725 respectively—a 4–5× difference. This within-abstract formulation isolates true reproducibility (same input, same conditions, different runs) from between-abstract content variation, providing a methodologically clean measure of field-level consistency.

E Prompt Card Example

The following is a complete, filled-in Prompt Card for the summarization task as used in our experiments:

Listing 3. Prompt Card for the scientific summarization task.

```
{
  "prompt_id": "summarization_v1",
  "prompt_hash": "c9644358805b4a7e...",
  "version": "1.0.0",
  "task_category": "summarization",
  "objective": "Produce a 3-sentence summary of a
    scientific abstract covering: (1) main
    contribution, (2) methodology, (3) key result.",
  "assumptions": [
    "Input is a single English scientific abstract",
    "Abstract contains identifiable methodology
     and quantitative results",
    "Model can produce coherent 3-sentence output"
  ],
  "limitations": [
    "Open-ended phrasing allows high output variance",
    "No explicit output-format constraint (unlike
     extraction task)"
  ],
  "target_models": [
    "llama3:8b", "mistral:7b", "gemma2:9b",
    "gpt-4", "claude-sonnet-4-5"
  ],
  "expected_output_format": "Three sentences of
    plain text, no JSON or structured markup",
  "interaction_regime": "single-turn",
  "change_log": [
    {"date": "2026-02-06", "change": "Initial version"}
  ]
}
```

F Representative Prompt Templates

The following are the exact prompt templates used for each of the four experimental tasks. In all templates, {`abstract`} is replaced with the scientific abstract text at runtime.

Task 1: Scientific Summarization

```
Summarize the following scientific abstract in exactly 3 sentences. Cover: (1) the main
contribution, (2) the methodology used, and (3) the key quantitative result.\n\nAbstract:
{abstract}\n\nSummary:
```

Task 2: Structured Extraction

```
Extract the following fields from the scientific abstract below. Return JSON only, no explanation.\n\nFields
objective, method, key_result, model_or_system, benchmark\n\nAbstract: {abstract}\n\nJSON:
```

Task 3: Multi-Turn Refinement (3 turns)

```
Turn 1: [Same as Task 1 prompt]\n Turn 2: Now revise the summary to be more specific about
the quantitative results mentioned.\n Turn 3: Finally, add one sentence about the limitations
or future work mentioned in the abstract.
```

Task 4: RAG Extraction

```
Using the context passage below and the scientific abstract, extract the following fields.
Return JSON only.\n\nContext: {retrieved_passage}\nAbstract: {abstract}\n\nFields: objective,
method, key_result, model_or_system, benchmark\n\nJSON:
```

G Experimental Coverage Matrix

Table 11 provides a complete coverage matrix showing the number of abstracts and runs per model–task–condition combination. This matrix enables readers to verify the sample sizes underlying all reported metrics.

H Chat-Format Control Experiment

To assess whether the prompt-format difference between LLaMA 3 (completion-style via `/api/generate`) and GPT-4 (chat-style via Chat Completions) contributes to the observed reproducibility gap, we conducted a supplementary control experiment running LLaMA 3 8B through Ollama's `/api/chat` endpoint, which applies the model's chat template (including special tokens for system/user/assistant roles) in the same message structure used by GPT-4.

Design: 10 abstracts $\times$ 2 tasks $\times$ 2 conditions (C1, C2) $\times$ 5 repetitions = 200 runs, all under greedy decoding (t=0).

Results: Table 12 compares the chat-format control with the original completion-format results for the same 10 abstracts. The two prompt formats produce *identical* variability metrics across all conditions: summarization EMR = 0.929, NED = 0.0066, and ROUGE-L = 0.9922 in both modes; extraction achieves perfect reproducibility (EMR = 1.000) regardless of interface. The 0.929 summarization EMR reflects the warm-up effect on 2 of 10 abstracts—the same pattern observed in the full 30-abstract experiment. These results confirm that prompt format is not a source of variability, and the reproducibility gap between LLaMA 3 and GPT-4 is consistent with deployment-side factors (server infrastructure, floating-point non-determinism across GPU types, request batching) rather than prompt-format differences.

I API Payload Documentation

To address potential "apples-to-oranges" concerns, we document the exact payload structures sent to each inference endpoint. All payloads were constructed deterministically and logged as part of the Run Card.

Local models (Ollama). Single-turn tasks use `POST /api/generate`:

Ollama generate payload (Tasks 1–2):

```
{"model": "llama3:8b",
 "prompt": "<full prompt text>",
 "options": {"temperature": 0.0, "seed": 42, "num_predict": 1024},
 "stream": false}
```

The `model` field is set to `llama3:8b`, `mistral:7b`, or `gemma2:9b` as appropriate. Multi-turn tasks (Task 3) use `POST /api/chat` with accumulated `messages` array. No system prompt, stop sequences, or post-processing are applied.

GPT-4 (OpenAI). Accessed via the `openai` Python SDK v1.59.9:

Table 11. Experimental coverage: number of abstracts (runs) per model–task–condition. Dash (–) indicates the combination was not evaluated. C1: fixed seed; C2: variable seed (C2_same_params for GPT-4); C3: temperature sweep at $t \in \{0.0, 0.3, 0.7\}$.

Model	Task	C1	C2	C3 (t=0.0)	C3 (t=0.3)	C3 (t=0.7)
LLaMA 3 8B	Extraction	30 (150)	30 (150)	30 (90)	30 (90)	30 (90)
	Summarization	30 (150)	30 (150)	30 (90)	30 (90)	30 (90)
	Multi-turn	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Mistral 7B	Extraction	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Summarization	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi-turn	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Gemma 2 9B	Extraction	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Summarization	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi-turn	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
GPT-4	Extraction	–	30 (150)	17 (51)	17 (51)	14 (42)
	Summarization	3 (8)†	30 (150)	30 (90)	30 (90)	30 (90)
	Multi-turn	–	–	–	–	–
	RAG	–	–	–	–	–
Claude Sonnet 4.5	Extraction	10 (49)‡	10 (50)	10 (30)	10 (30)	10 (30)
	Summarization	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi-turn	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Gemini 2.5 Pro	Multi-turn	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
DeepSeek Chat	Extraction	10 (50)	–	–	–	–
	Summarization	10 (50)	–	–	–	–
Perplexity Sonar	Extraction	10 (50)	–	–	–	–
	Summarization	10 (50)	–	–	–	–
Together AI LLaMA 3 8B	Extraction	10 (50)	10 (50)	–	–	–
	Summarization	10 (50)	10 (50)	–	–	–

†GPT-4 C1 summarization: only 3 abstracts completed before quota exhaustion; excluded from primary analysis (C2 used instead).
‡Claude C1 extraction: 49 runs (1 empty output due to API timeout).
§An additional 200 runs (LLaMA 3 8B, chat-format control; Appendix H) are not shown in this table.

```
{"model": "gpt-4",
 "messages": [{"role": "user", "content": "<prompt>"}],
 "temperature": 0.0, "seed": 42, "max_tokens": 1024}
```

No system message, stop sequences, `top_p`, `frequency_penalty`, or `presence_penalty` were set (all defaults). The resolved model version (`gpt-4-0613`) was extracted from the response object and logged.

Claude Sonnet 4.5 (Anthropic). Accessed via `urllib` (no SDK dependency):

Table 12. Prompt-format control: LLaMA 3 8B via completion (`/api/generate`) vs. chat (`/api/chat`) for 10 abstracts under greedy decoding (t=0). EMR computed over conditions C1 and C2 combined.

Task	Metric	Completion	Chat
Summarization	EMR↑	0.929	0.929
	NED↓	0.0066	0.0066
	ROUGE-L↑	0.9922	0.9922
Extraction	EMR↑	1.000	1.000
	NED↓	0.0000	0.0000
	ROUGE-L↑	1.0000	1.0000

Note: Completion and chat formats yield identical metrics for all 10 abstracts under greedy decoding, indicating that prompt format is not a source of variability.

```
{"model": "claude-sonnet-4-5-20250929",
 "messages": [{"role": "user", "content": "<prompt>"}],
 "temperature": 0.0, "max_tokens": 1024}
```

No `seed` parameter (not supported by the Anthropic API), no system message, no stop sequences. The seed value in the Run Card is marked `seed_status: "logged-only-not-sent-to-api"`.

Gemini 2.5 Pro (Google). Accessed via `urllib` (no SDK dependency) through the Google AI Studio REST API:

```
{"contents": [{"role": "user", "parts": [{"text": "<prompt>"}]}],
 "generationConfig": {"maxOutputTokens": 8192,
  "temperature": 0.0, "seed": 42},
 "systemInstruction": {"parts": [{"text": "<system>"}]}}
```

The `seed` parameter is supported by the Gemini API and is sent with every request. Gemini 2.5 Pro is a "thinking" model: internal reasoning tokens count against the `maxOutputTokens` budget, hence the higher limit (8,192 vs. 1,024 for other models). Multi-turn tasks use the same endpoint with accumulated `contents` array alternating `role: "user"` and `role: "model"`.

DeepSeek Chat (DeepSeek). Accessed via the OpenAI-compatible API:

```
{"model": "deepseek-chat",
 "messages": [{"role": "user", "content": "<prompt>"}],
 "temperature": 0.0, "max_tokens": 1024}
```

No seed parameter, system message, or stop sequences. DeepSeek Chat achieved the highest API reproducibility (EMR = 0.800 for extraction), suggesting effective internal determinism under greedy decoding.

Perplexity Sonar (Perplexity). Accessed via the Perplexity API:

```
{"model": "sonar",
 "messages": [{"role": "user", "content": "<prompt>"}],
 "temperature": 0.0, "max_tokens": 1024}
```

Perplexity Sonar is an online model with real-time search augmentation. No seed parameter or system message. The search-augmented nature introduces an additional source of variability: retrieved web content may differ across requests, contributing to the lowest observed reproducibility (EMR = 0.010–0.100).

Key symmetry points. Across all nine model deployments: (1) identical prompt text (verified by `prompt_hash`); (2) identical temperature (t=0.0); (3) identical max token limit (1,024; 8,192 for Gemini

2.5 Pro to accommodate thinking tokens); (4) no system messages for single-turn tasks; (5) no stop sequences; (6) no post-processing or text normalization of outputs.

J JAIR Reproducibility Compliance

This appendix documents compliance with the JAIR reproducibility mechanisms described by **gundersen2024improving**. Each item maps to the four mechanisms: checklists, structured abstracts, badges, and reproducibility reports.

Mechanism 1: Reproducibility Checklist

✓ **Code availability**: Reference implementation publicly available at https://github.com/Roverlucas/genai-reproducibility-protocol (MIT License).

✓ **Data availability**: All 4,104 run records, PROV documents, and 30 input abstracts included in the repository.

✓ **Experimental methodology**: Full description of models, tasks, conditions, and metrics in Section 4.

✓ **Hyperparameters**: All inference parameters documented per model (Appendix I); temperature = 0, max_tokens = 1024 across all models.

✓ **Statistical tests**: Holm-Bonferroni correction across 68 tests, Fisher's exact tests, bootstrap 95% CIs (10,000 resamples).

✓ **Randomness control**: Random seeds documented; bootstrap seed = 42; five experimental conditions (C1, C2, C3 with three temperature sub-conditions) isolate seed and parameter effects.

✓ **Computing infrastructure**: Hardware and software environment documented (Apple M4, Ollama v0.15.5, Python 3.14).

✓ **Run time**: Protocol overhead measured at <1% of inference time (≈4 KB per run).

Mechanism 2: Structured Abstract

✓ Abstract follows JAIR's structured format: Background, Objectives, Methods, Results, Conclusions.

Mechanism 3: Reproducibility Badge Eligibility

✓ **Code**: Complete reference implementation in Python.

✓ **Data**: All experimental data (run records, provenance graphs, input texts) publicly available.

✓ **Experiment**: Full pipeline (data collection, analysis, figure generation) can be re-executed from the repository.

Mechanism 4: Reproducibility Report

✓ Sensitivity analysis via balanced 10-abstract subsample confirms findings (Section 5.1).

✓ Chat-format control experiment (200 additional runs) rules out prompt-format confound (Appendix H).

✓ Automated test suite (51 tests) validates core analysis functions.

✓ Continuous integration pipeline ensures reproducibility of code and analysis.

Received February 2026