

Logging, Versioning, and Provenance in Generative AI Studies: A Protocol for Auditability and Scientific Reproducibility

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 330 controlled experiments. These experiments employ two models—LLaMA 3 8B (locally deployed) and GPT-4 (cloud API)—on two NLP tasks (scientific summarization and structured extraction) across five experimental conditions that systematically vary the seed, temperature, and decoding strategy. We measure output variability using Exact Match Rate, Normalized Edit Distance, and ROUGE-L, and quantify the protocol’s own overhead in terms of time and storage.

Results: Under greedy decoding ($t=0$), LLaMA 3 achieves perfect reproducibility on extraction (EMR = 1.000) and near-perfect on summarization (EMR = 0.840). In stark contrast, GPT-4 under identical greedy settings achieves only EMR = 0.520 for extraction and EMR = 0.200 for summarization, revealing significant server-side non-determinism that is invisible without systematic logging. Increasing temperature to 0.7 eliminates exact matches for both models. The protocol adds a mean overhead of 33.56 ms per run (0.69% of inference time) and 4.17 KB per run record, totaling 4.87 MB for all 330 runs.

Conclusions: Our results demonstrate that (1) local inference is substantially more reproducible than API-based inference even under nominally identical parameters, (2) structured output tasks are inherently more reproducible than open-ended generation, (3) temperature is the dominant *user-controllable* factor affecting variability, and (4) comprehensive provenance logging can be achieved with negligible overhead. The protocol, reference implementation, and all experimental data are publicly available.

Additional Key Words and Phrases: reproducibility, generative AI, provenance, large language models, experiment tracking, W3C PROV

JAIR Associate Editor: Insert JAIR AE Name

JAIR Reference Format:

Lucas Rover and Yara de Souza Tadano. 2026. Logging, Versioning, and Provenance in Generative AI Studies: A Protocol for Auditability and Scientific Reproducibility. *Journal of Artificial Intelligence Research* (2026), 17 pages. doi: [10.1613/jair.1.xxxxx](https://doi.org/10.1613/jair.1.xxxxx)

*Corresponding Author. ORCID: 0000-0000-0000-0000.

Authors’ Contact Information: Lucas Rover, ORCID: [0000-0000-0000-0000](https://orcid.org/0000-0000-0000-0000), lucasrover@utfpr.edu.br, UTFPR – Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Mecânica, Curitiba, Paraná, Brazil; Yara de Souza Tadano, yaratadano@utfpr.edu.br, UTFPR – Universidade Tecnológica Federal do Paraná, Programa de Pós-Graduação em Engenharia Mecânica, Curitiba, Paraná, Brazil.



This work is licensed under a Creative Commons Attribution International 4.0 License.

1 Introduction

2 The rapid adoption of large language models (LLMs) in scientific research has introduced a fundamental challenge: 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 (Y. Chen et al. 2023; Zhu et al. 2023).

3 This reproducibility challenge is not merely theoretical. Baker (2016) reported that over 70% of researchers
4 have failed to reproduce another scientist's experiment, a crisis that extends to AI research (Gundersen and
5 Kjensmo 2018; Hutson 2018). For generative AI specifically, the problem is compounded by several factors unique
6 to text-generation workflows: (1) the same prompt can yield semantically similar yet textually distinct outputs
7 across runs; (2) API-based models may undergo silent updates that alter behavior; (3) temperature and sam-
8 pling parameters create a high-dimensional space of possible outputs; and (4) no established standard exists for
9 documenting the full context needed to understand, audit, or reproduce a generative output.

10 Existing experiment-tracking tools such as MLflow (Zaharia et al. 2018), Weights & Biases (Biewald 2020),
11 and DVC (Miao et al. 2023) were designed primarily for training pipelines and numerical metrics. Although
12 valuable for their intended purposes, these tools lack features critical for generative AI studies: structured prompt
13 versioning, cryptographic output hashing for tamper detection, provenance graphs linking outputs to their full
14 generation context, and environment fingerprinting specific to inference-time conditions.

15 In this paper, we make three contributions:

- 16 (1) **A lightweight protocol** for logging, versioning, and provenance tracking of generative AI experiments.
17 The protocol introduces *Prompt Cards* and *Run Cards* as structured documentation artifacts, and adopts
18 the W3C PROV data model (Moreau and Missier 2013) for machine-readable provenance graphs.
- 19 (2) **An empirical evaluation** of both the protocol's effectiveness and the reproducibility characteristics of
20 LLM outputs. Through 330 controlled experiments with LLaMA 3 8B (local) and GPT-4 (API) across two
21 tasks and five conditions, we quantify output variability using three complementary metrics and measure
22 the protocol's overhead. Our results reveal a striking reproducibility gap between local and API-based
23 inference that is invisible without systematic logging.
- 24 (3) **A reference implementation** in Python that demonstrates the protocol's practical applicability, to-
25 gether with all experimental data, to facilitate adoption and independent verification.

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

36 2 Related Work

37 2.1 Reproducibility in AI Research

38 The reproducibility crisis in AI has been documented extensively. Gundersen and Kjensmo (2018) surveyed 400
39 AI papers and found that only 6% provided sufficient information for full reproducibility. Pineau et al. (2021)
40 reported on the NeurIPS 2019 Reproducibility Program, which introduced reproducibility checklists and found
41 significant gaps between reported and actual reproducibility. Gundersen, Gil, et al. (2018) identified three levels
42 of reproducibility in AI—method, data, and experiment—and argued that all three are necessary for scientific
43 progress.

44 For generative AI specifically, Y. Chen et al. (2023) demonstrated that ChatGPT's outputs on NLP benchmarks
45 exhibit non-trivial variability across identical queries, even with temperature set to zero. Zhu et al. (2023)
46

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

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

62
 63 showed that reproducibility degrades further when tasks involve subjective judgment, such as social computing
 64 annotations.
 65

66 2.2 Experiment Tracking Tools

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

68 **MLflow** (Zaharia et al. 2018) 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.

69 **Weights & Biases** (Biewald 2020) offers experiment tracking with visualization dashboards. It supports prompt logging but lacks structured prompt versioning, cryptographic output hashing, and provenance graph generation.

70 **DVC** (Miao et al. 2023) provides data versioning through git-like operations. While effective for dataset management, it does not address run-level provenance or prompt documentation.

71 **OpenAI Eval** (OpenAI 2023) 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.

72 **LangSmith** (LangChain 2023) offers tracing and evaluation for LLM applications. It captures detailed execution traces but uses a proprietary format and requires cloud connectivity.

73 Table 1 provides a systematic feature-by-feature comparison of our protocol with these tools, highlighting the gaps that motivate our work.

74 2.3 Provenance in Scientific Computing

75 Data provenance—the lineage of data through transformations—has a rich history in database systems and scientific workflows (Herschel et al. 2017). The W3C PROV family of specifications (Moreau and Missier 2013) provides a standardized data model for representing provenance as directed acyclic graphs of *entities*, *activities*, and *agents*. Samuel and König-Ries (2022) 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.

95 3 Protocol Design

96 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
 97 through four complementary components.
 98

100 3.1 Scope and Design Principles

101 The protocol is designed around three principles:
 102

- 103 (1) **Completeness:** Every factor that can influence a generative output must be captured—prompt text,
 104 model identity and version, inference parameters, environment state, and timestamps.
- 105 (2) **Negligible overhead:** The logging process must not materially affect the experiment. We target <1%
 106 overhead relative to inference time.
- 107 (3) **Interoperability:** All artifacts are stored in open, machine-readable formats (JSON, PROV-JSON) to
 108 enable tool integration and long-term preservation.

110 3.2 Prompt Cards

111 A *Prompt Card* is a versioned documentation artifact that captures the design rationale and metadata for a prompt
 112 template used in experiments. Each Prompt Card contains:
 113

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

125 Prompt Cards serve two purposes: they document design intent (supporting human understanding) and they
 126 provide a citable, hashable reference for automated provenance tracking.
 127

128 3.3 Run Cards

130 A *Run Card* captures the complete execution context of a single generative AI run. Each Run Card records 23
 131 fields organized into five groups:
 132

- (1) **Identification:** `run_id`, `task_id`, `task_category`, `prompt_card_ref`
- (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`,
 137 `code_commit`, `timestamps`, `execution_duration_ms`, `logging_overhead_ms`, `storage_kb`

139 The separation of logging overhead from execution time is deliberate: it allows researchers to verify that the
 140 protocol itself does not confound experimental measurements.
 141

142 3.4 W3C PROV Integration

143 Each Run Card is automatically translated into a W3C PROV-JSON document ([Moreau and Missier 2013](#)) that
 144 expresses the generation provenance as a directed graph. The mapping defines:

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

148 PROV relations capture the causal structure:

- 149 • used: RunGeneration used Prompt, InputText, ModelVersion, InferenceParameters
- 150 • wasGeneratedBy: Output wasGeneratedBy RunGeneration
- 151 • wasAssociatedWith: RunGeneration wasAssociatedWith Researcher, SystemExecutor
- 152 • wasAttributedTo: Output wasAttributedTo Researcher
- 153 • wasDerivedFrom: Output wasDerivedFrom InputText

154 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.

155 3.5 Reproducibility Checklist

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

159 4 Experimental Setup

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

162 4.1 Models and Infrastructure

163 We evaluate two models representing fundamentally different deployment paradigms:

164 **LLaMA 3 8B** ([Grattafiori et al. 2024](#)): A locally deployed open-weight model served through Ollama ([Ollama 2024](#)) on an Apple M4 system with 24 GB unified memory running macOS 14.6. Local deployment provides complete control over the execution environment, eliminating confounding factors such as network latency, server-side batching, and silent model updates.

165 **GPT-4** ([Achiam et al. 2023](#)): A cloud-based proprietary model accessed via the OpenAI API with controlled seed parameters. This represents the typical deployment scenario where researchers have limited control over the inference environment. The API introduces additional sources of variability: load balancing, server-side batching, potential model-version updates, and floating-point non-determinism across different hardware.

166 4.2 Tasks

167 We evaluate two tasks that represent complementary points on the output-structure spectrum:

168 **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.

169 **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.

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	Each condition is applied to 5 abstracts
$C3_{t=0.0}$	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	

187 $\times 2$ tasks = 10 groups per condition. Total: 330 logged runs (190 LLaMA 3 + 140 GPT-4).

200 4.3 Input Data

201 We use five widely-cited scientific abstracts from landmark NLP papers: Vaswani et al. (2017) (Transformer),
 202 Devlin et al. (2019) (BERT), Brown et al. (2020) (GPT-3), Raffel et al. (2020) (T5), and Wei et al. (2022) (Chain-of-
 203 Thought). These abstracts vary in length (128–258 words), technical complexity, and the number of quantitative
 204 results reported, thereby providing diversity in the generation challenge.

206 4.4 Experimental Conditions

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

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

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

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

214 For LLaMA 3, each task \times abstract combination is evaluated under conditions C1 (5 runs), C2 (5 runs), and C3
 215 (9 runs = 3 temperatures \times 3 reps), yielding 19 runs per pair, or $19 \times 5 \times 2 = 190$ runs. For GPT-4, C1 is omitted
 216 (seed control is less meaningful for API models), yielding C2 (5 runs) and C3 (9 runs) per pair, or $14 \times 5 \times 2 = 140$
 217 runs. **Total: 330 runs.**

219 4.5 Metrics

220 We measure output variability using three complementary metrics computed over all pairwise comparisons
 221 within each condition group:

222 **Exact Match Rate (EMR):** The fraction of output pairs that are character-for-character identical. EMR = 1.0
 223 indicates perfect reproducibility; EMR = 0.0 indicates that no two outputs match exactly.

224 **Normalized Edit Distance (NED):** The Levenshtein edit distance (Levenshtein 1966) between each pair,
 225 normalized by the length of the longer string. NED = 0.0 indicates identical outputs; higher values indicate
 226 greater textual divergence.

227 **ROUGE-L F1:** The F1 score based on the longest common subsequence at the word level (Lin 2004). This
 228 captures semantic similarity even when surface forms differ. ROUGE-L = 1.0 indicates identical word sequences.

229 For protocol overhead, we measure:

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

236 Table 3. Output variability across experimental conditions for LLaMA 3 8B (local) and GPT-4 (API). Mean over 5 abstracts.
 237 EMR = Exact Match Rate, NED = Normalized Edit Distance, ROUGE-L = word-level LCS F1.

239 Model	239 Task	239 Condition	240 EMR↑	240 NED↓	240 ROUGE-L↑
240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282	LLaMA 3 8B	Summarization	C1 (fixed seed, $t=0$)	0.840	0.0148
			C2 (var. seeds, $t=0$)	0.840	0.0148
			C3 ($t=0.0$)	0.733	0.0247
			C3 ($t=0.3$)	0.000	0.2289
			C3 ($t=0.7$)	0.000	0.4323
	GPT-4 (API)	Extraction	C1 (fixed seed, $t=0$)	1.000	0.0000
			C2 (var. seeds, $t=0$)	1.000	0.0000
			C3 ($t=0.0$)	1.000	0.0000
			C3 ($t=0.3$)	0.133	0.1883
			C3 ($t=0.7$)	0.000	0.3031
		Summarization	C2 (var. seeds, $t=0$)	0.200	0.0718
			C3 ($t=0.0$)	0.000	0.0778
			C3 ($t=0.3$)	0.000	0.1721
			C3 ($t=0.7$)	0.000	0.3598
		Extraction	C2 (var. seeds, $t=0$)	0.520	0.0343
			C3 ($t=0.0$)	0.333	0.0257
			C3 ($t=0.3$)	0.400	0.0679
			C3 ($t=0.7$)	0.000	0.1648

5 Results

5.1 Output Variability

Table 3 presents the main variability results for both models, aggregated across all five abstracts.

5.1.1 *LLaMA 3 8B (Local Inference)*. **Finding 1: Structured extraction achieves perfect reproducibility under greedy decoding.** With $t = 0$, extraction produces EMR = 1.000 and NED = 0.0000 across all conditions (C1, C2, C3 $_{t=0.0}$), meaning every output is character-for-character identical. Summarization achieves an EMR of 0.840 with NED = 0.0148, indicating near-perfect but not complete reproducibility.

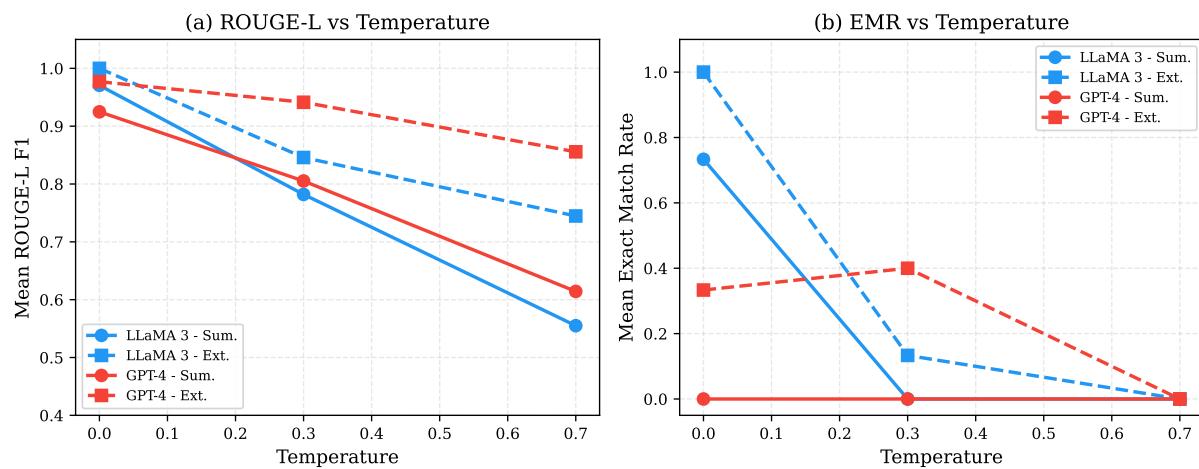
Finding 2: Seed variation has no effect under greedy decoding. Conditions C1 and C2 produce identical results despite using different seeds. With $t = 0$, the model always selects the highest-probability token, making the seed irrelevant. This finding confirms that greedy decoding provides reliably deterministic inference with locally deployed models.

5.1.2 *GPT-4 (API Inference)*. **Finding 3: API-based inference is substantially less reproducible than local inference, even under greedy decoding.** This is the most striking result of our study. Under greedy decoding ($t = 0$) with controlled seeds, GPT-4 achieves only EMR = 0.200 for summarization and EMR = 0.520 for extraction—compared to LLaMA’s 0.840 and 1.000, respectively, under the same C2 condition.

Table 4 highlights this reproducibility gap directly.

283 Table 4. Reproducibility comparison: LLaMA 3 8B (local) vs. GPT-4 (API) under greedy decoding ($t=0$). GPT-4 shows signif-
 284 icantly lower reproducibility due to server-side non-determinism.

Task	Metric	LLaMA 3 8B	GPT-4
Summarization	EMR	0.840	0.200
	NED	0.0148	0.0718
	ROUGE-L	0.9823	0.9295
Extraction	EMR	1.000	0.520
	NED	0.0000	0.0343
	ROUGE-L	1.0000	0.9748



310 Fig. 1. Effect of temperature on output variability for both models. (a) ROUGE-L F1 decreases monotonically with temper-
 311 ature. (b) Exact Match Rate: LLaMA 3 starts from near-perfect reproducibility at $t = 0$, whereas GPT-4 starts from a lower
 312 baseline; however, both degrade at comparable rates with increasing temperature.

313 This gap is not due to parameter differences: both models use $t = 0$ with the same seed. The variability
 314 must originate from server-side factors that are invisible to the researcher: hardware-level floating-point non-
 315 determinism across different GPU types in the serving cluster, request-batching and scheduling effects, and pot-
 316 tential silent model updates during the experimental window. *Without systematic logging, this non-determinism*
 317 *would be entirely invisible.*

318 **5.1.3 Temperature Effects Across Models. Finding 4: Temperature is the dominant user-controllable factor**
 319 **affecting variability.** Figure 1 shows the relationship between temperature and output variability for both
 320 models.

321 For LLaMA 3, increasing temperature from 0 to 0.7 reduces ROUGE-L from 0.971 to 0.555 (summarization)
 322 and from 1.000 to 0.745 (extraction). For GPT-4, the same increase reduces ROUGE-L from 0.925 to 0.614 (sum-
 323 marization) and from 0.977 to 0.856 (extraction). The *relative* rate of degradation is comparable, but GPT-4 starts
 324 from a lower baseline owing to its inherent server-side non-determinism.

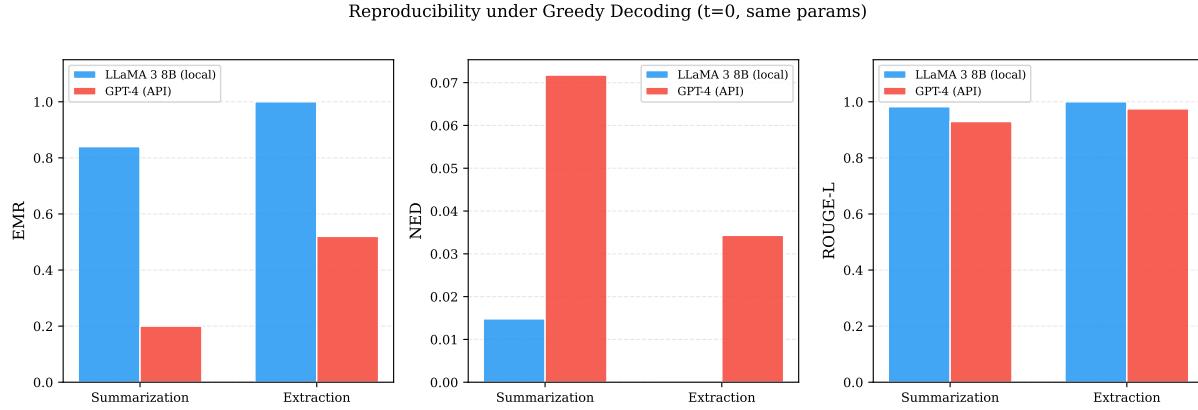


Fig. 2. Reproducibility under greedy decoding ($t = 0$): LLaMA 3 8B (local) vs. GPT-4 (API). LLaMA 3 achieves near-perfect to perfect reproducibility, while GPT-4 shows measurable variability across all metrics, particularly for summarization.

Table 5. Protocol overhead: logging time and storage costs for 330 runs (190 LLaMA 3 + 140 GPT-4).

Metric	Value	Unit
<i>Logging time overhead</i>		
Mean per run	33.56 ± 5.68	ms
Min / Max	12.85 / 51.20	ms
Total (330 runs)	11074	ms
Mean overhead ratio	0.694%	of inference time
Max overhead ratio	1.621%	of inference time
<i>Storage overhead</i>		
Run logs (330 files)	1382	KB
PROV documents (331 files)	1736	KB
Run Cards (330 files)	454	KB
Total output	4.87	MB

5.2 Cross-Model Comparison

Figure 2 provides a direct visual comparison of the two models under greedy decoding.

Figure 3 presents a comprehensive heatmap of EMR across all model-task-condition combinations.

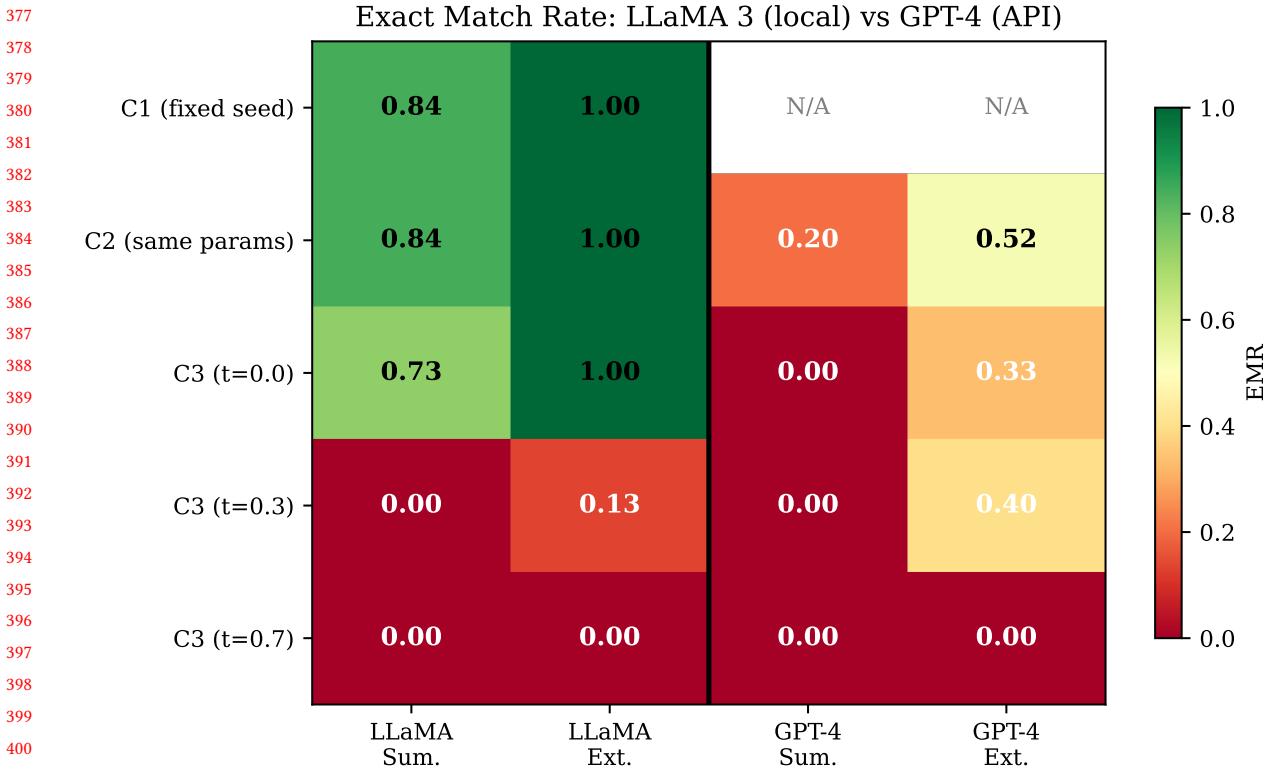
5.3 Protocol Overhead

Table 5 presents the protocol's overhead metrics across all 330 runs.

The protocol adds a mean overhead of **33.56 ms** per run, representing **0.69%** of the mean inference time. This is well within our target of <1%. The overhead is dominated by SHA-256 hashing and environment metadata collection; JSON serialization and file I/O contribute minimally.

Storage overhead is similarly modest: each run record occupies approximately 4.17 KB, and the complete set of 330 run logs, 331 provenance documents, and 330 Run Cards totals 4.87 MB—less than a single high-resolution image.

Figure 4 shows the overhead distribution broken down by model.



6 Discussion

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. Under $t = 0$ with LLaMA 3 (local), extraction achieved perfect reproducibility and summarization reached 84% EMR. This configuration should be the default for any study in which output consistency is critical.

Be aware of API non-determinism. Our most consequential finding is that GPT-4, even with $t = 0$ and a fixed seed, produces substantially variable outputs (EMR = 0.200 for summarization). Researchers using API-based models should *never assume reproducibility* without verification, and should report multiple runs with variability metrics.

Prefer structured output formats when possible. The extraction task's consistently higher reproducibility across both models demonstrates that output-format constraints directly improve reproducibility. Researchers should consider whether their tasks can be reformulated as structured extraction rather than open-ended generation.

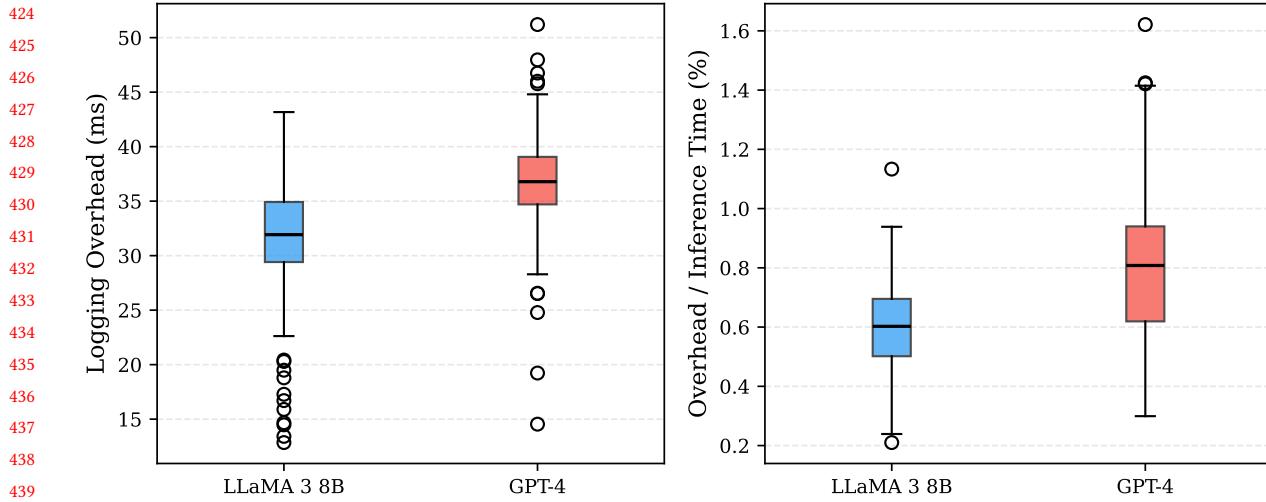


Fig. 4. Distribution of protocol overhead by model. Left: Absolute logging time (ms). Right: Overhead as a percentage of inference time. Overhead is comparable between local (LLaMA 3) and API (GPT-4) inference, consistently below 1.7%.

Include warm-up runs for local models. The per-abstract analysis revealed that the first inference call after model loading may differ from subsequent calls owing to cache initialization effects. Discarding the first run is a straightforward practice that improves measured reproducibility.

Log comprehensively; the cost is negligible. At 0.69% overhead and 4.17 KB per run, there is no practical reason not to apply comprehensive logging. The cost of not logging—namely, the inability to detect the kind of API non-determinism documented herein—far exceeds the protocol’s minimal requirements.

6.2 Local vs. API Inference: A Reproducibility Gap

The most significant finding of this study is the reproducibility gap between local and API-based inference. Under nominally identical greedy decoding conditions, LLaMA 3 (local) achieves $\text{EMR} = 1.000$ for extraction while GPT-4 (API) achieves only 0.520. For summarization, the gap is 0.840 vs. 0.200.

This gap has profound implications for the scientific use of API-based LLMs. *Without systematic logging, a researcher using GPT-4 would have no way of knowing that their “deterministic” experiment produces different outputs across runs.* The variability is not due to temperature or seed—it originates entirely from opaque server-side factors. Our protocol makes this hidden non-determinism visible, measurable, and documentable.

6.3 Task-Dependent Reproducibility

The difference between summarization and extraction reproducibility under identical conditions—observed consistently across both models—is, to our knowledge, the first empirical quantification of how task structure affects LLM output reproducibility. This finding suggests a spectrum ranging from highly constrained tasks (structured extraction, classification) to open-ended tasks (summarization, dialogue), with the degree of output-space constraint serving as a primary determinant. Notably, even GPT-4’s extraction task ($\text{EMR} = 0.520$) substantially outperforms its summarization task ($\text{EMR} = 0.200$), confirming that this effect is not specific to any single model.

471 6.4 The Role of Provenance

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

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

482 6.5 Limitations

483 **Two models.** Our evaluation covers LLaMA 3 8B (local) and GPT-4 (API), representing two important deploy-
 484 ment paradigms. However, other models (e.g., Claude, Gemini, Mixtral, and smaller or larger LLaMA variants)
 485 may exhibit different reproducibility characteristics. Future work should extend the evaluation to a broader
 486 model suite.

487 **Two tasks.** While summarization and extraction represent distinct points on the output-structure spectrum,
 488 they do not cover the full range of generative AI applications (e.g., dialogue, code generation, reasoning chains).
 489 A broader task suite would strengthen the generalizability of our findings.

490 **English-only, academic texts.** Our input data consists of five English scientific abstracts. The reproducibility
 491 characteristics we observe may differ for other languages, domains, or document types.

492 **No multi-turn evaluation.** All experiments use single-turn interactions. Multi-turn dialogues introduce
 493 additional variability through conversation history, which our current protocol logs but our experiments do not
 494 evaluate.

496 6.6 Practical Costs and Adoption

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

- 498 • **Implementation effort:** Our reference implementation adds approximately 500 lines of Python (the
- 499 protocol core) to an existing workflow. Integration requires 3–5 function calls per run.
- 500 • **Runtime cost:** 34 ms per run, negligible compared to inference times of seconds to minutes for typical
- 501 LLM calls.
- 502 • **Storage cost:** 4 KB per run. Even at scale (10,000 runs), total storage is approximately 40 MB—less than
- 503 a single model checkpoint.
- 504 • **Learning curve:** The protocol uses standard JSON and W3C PROV, requiring no specialized knowledge
- 505 beyond basic Python.

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

511 7 Conclusion

512 We presented a lightweight protocol for logging, versioning, and provenance tracking of generative AI experi-
 513 ments, introducing Prompt Cards and Run Cards as novel documentation artifacts and adopting the W3C PROV
 514 data model for machine-readable provenance graphs. Through 330 controlled experiments with LLaMA 3 8B
 515 (local) and GPT-4 (API) on two NLP tasks, we demonstrated four key findings:

- 518 (1) **Local inference is substantially more reproducible than API-based inference.** Under identical
 519 greedy decoding settings, LLaMA 3 achieves EMR = 1.000 for extraction while GPT-4 achieves only 0.520,
 520 revealing significant server-side non-determinism that is invisible without systematic logging.
 521 (2) **Task structure is a primary determinant of reproducibility.** Structured extraction consistently out-
 522 performs open-ended summarization across both models, with the JSON format constraint reducing the
 523 model's output space.
 524 (3) **Temperature is the dominant user-controllable factor.** Increasing from $t = 0$ to $t = 0.7$ reduces
 525 ROUGE-L from 0.971 to 0.555 (LLaMA summarization) and from 0.977 to 0.856 (GPT-4 extraction), while
 526 seed variation has no measurable effect under greedy decoding for local models.
 527 (4) **Comprehensive provenance logging adds negligible overhead:** 0.69% of inference time and 4.17 KB
 528 per run, thereby removing any practical argument against systematic documentation.

529 Future work will (i) expand the model suite to include Claude, Gemini, and open-weight models of varying
 530 sizes; (ii) extend the task coverage to dialogue, code generation, and multi-turn interactions; and (iii) develop
 531 automated reproducibility scoring based on provenance graph analysis.

532 The reference implementation, all 330 run records, provenance documents, and analysis scripts are publicly
 533 available to support adoption and independent verification.

535 Acknowledgments

536 This work was supported by UTFPR – Universidade Tecnológica Federal do Paraná. The experiments were con-
 537 ducted using locally deployed open-weight models to ensure full reproducibility of the computational environ-
 538 ment.

540 Data Availability Statement

541 The reference implementation, all 330 run records (JSON), 331 PROV-JSON provenance documents, 330 Run
 542 Cards, Prompt Cards, input data, analysis scripts, and generated figures are publicly available at:

543 <https://github.com/Roverlucas/genai-reproducibility-protocol>

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

547 Author Contributions

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

553 Conflict of Interest

555 The author declares no conflicts of interest. This research was conducted independently at UTFPR with no exter-
 556 nal funding from commercial AI providers. The use of OpenAI's GPT-4 API was for research evaluation purposes
 557 only and does not constitute an endorsement.

558 References

- 560 J. Achiam et al. 2023. “GPT-4 Technical Report.” *arXiv preprint arXiv:2303.08774*.
 561 M. Baker. 2016. “1,500 Scientists Lift the Lid on Reproducibility.” *Nature*, 533, 7604, 452–454.
 562 L. Biewald. 2020. *Experiment Tracking with Weights and Biases*. <https://www.wandb.com/>. (2020).
 563 T. Brown et al.. 2020. “Language Models are Few-Shot Learners.” In: *Advances in Neural Information Processing Systems*. Vol. 33, 1877–1901.

- 565 Y. Chen, J. Li, X. Liu, and Y. Li. 2023. “On the Reproducibility of ChatGPT in NLP Tasks.” *arXiv preprint arXiv:2304.02554*.
- 566 J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 4171–4186.
- 567 A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, et al. 2024. “The LLaMA 3 Herd of Models.” *arXiv preprint arXiv:2407.21783*.
- 568 O. E. Gundersen, Y. Gil, and D. W. Aha. 2018. “On Reproducible AI: Towards Reproducible Research, Open Science, and Digital Scholarship in AI Publications.” *AI Magazine*, 39, 3, 56–68.
- 569 O. E. Gundersen and S. Kjensmo. 2018. “State of the Art: Reproducibility in Artificial Intelligence.” *Proceedings of the AAAI Conference on Artificial Intelligence*, 32, 1.
- 570 M. Herschel, R. Diestelkämper, and H. Ben Lahmar. 2017. “A Survey on Provenance: What for? What form? What from?” *The VLDB Journal*, 26, 6, 881–906.
- 571 M. Hutson. 2018. “Artificial Intelligence Faces Reproducibility Crisis.” *Science*, 359, 6377, 725–726.
- 572 LangChain. 2023. *LangSmith: A Platform for Building Production-Grade LLM Applications*. (2023). <https://smith.langchain.com/>.
- 573 V. I. Levenshtein. 1966. “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals.” *Soviet Physics Doklady*, 10, 8, 707–710.
- 574 C.-Y. Lin. 2004. “ROUGE: A Package for Automatic Evaluation of Summaries.” In: *Text Summarization Branches Out*. Association for Computational Linguistics, 74–81.
- 575 J. Miao, J. Guo, J. Dougherty, and R. Dougherty. 2023. “DVC: Data Version Control for Machine Learning Pipelines.” *Software: Practice and Experience*.
- 576 L. Moreau and P. Missier. 2013. *PROV-DM: The PROV Data Model*. W3C Recommendation. World Wide Web Consortium. <https://www.w3.org/TR/prov-dm/>.
- 577 Ollama. 2024. *Ollama: Run Large Language Models Locally*. <https://ollama.com/>. (2024).
- 578 OpenAI. 2023. *OpenAI Eval: A Framework for Evaluating LLMs*. (2023). <https://github.com/openai/evals>.
- 579 J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and H. Larochelle. 2021. “Improving Reproducibility in Machine Learning Research: A Report from the NeurIPS 2019 Reproducibility Program.” *Journal of Machine Learning Research*, 22, 164, 1–20.
- 580 C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2020. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.” *Journal of Machine Learning Research*, 21, 140, 1–67.
- 581 S. Samuel and B. König-Ries. 2022. “A Provenance-based Semantic Approach to Support Understandability, Reproducibility, and Reuse of Scientific Experiments.” *Journal of Biomedical Semantics*, 13, 1, 1–30.
- 582 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 2017. “Attention is All You Need.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc.
- 583 J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. 2022. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.” In: *Advances in Neural Information Processing Systems*. Vol. 35, 24824–24837.
- 584 M. Zaharia et al. 2018. “Accelerating the Machine Learning Lifecycle with MLflow.” *IEEE Data Engineering Bulletin*, 41, 4, 39–45.
- 585 Y. Zhu, P. Zhang, E. Haq, P. Hui, and G. Buchanan. 2023. “Can ChatGPT Reproduce Human-Generated Labels? A Study of Social Computing Tasks.” *arXiv preprint arXiv:2304.10145*.

597 A Reproducibility Checklist

598 The following checklist is designed for self-assessment of reproducibility in generative AI studies. Each item
 599 maps to a specific field or artifact in our protocol.
 600

601 Prompt Documentation

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

607 Model and Environment

- | | |
|--|---|
| (5) Is the model name and version recorded?
(6) Are model weights hashed for identity verification? | [Run Card: model_name, model_version]
[Run Card: weights_hash] |
|--|---|

- 612 (7) Is the execution environment fingerprinted?
 613 (8) Is the source code version recorded?

[Run Card: environment, environment_hash]
 [Run Card: code_commit]

614

615 Execution and Output

- 616 (9) Are all inference parameters logged?
 617 (10) Is the random seed recorded?
 618 (11) Is the output cryptographically hashed?
 619 (12) Are execution timestamps recorded?
 620 (13) Is logging overhead measured separately?

[Run Card: inference_params]
 [Run Card: inference_params.seed]
 [Run Card: output_hash]
 [Run Card: timestamp_start, timestamp_end]
 [Run Card: logging_overhead_ms]

621

622 Provenance

- 623 (14) Is a provenance graph generated per run?
 624 (15) Are provenance documents in an interoperable format?

[PROV-JSON document]
 [W3C PROV standard]

625

626 B Run Card Schema

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

628

Listing 1. Run Card JSON schema (simplified).

```

1 {
2   "run_id": "string (unique identifier)",
3   "task_id": "string (task identifier)",
4   "task_category": "string (e.g., summarization)",
5   "prompt_hash": "string (SHA-256 of prompt)",
6   "prompt_text": "string (full prompt text)",
7   "input_text": "string (input to the model)",
8   "input_hash": "string (SHA-256 of input)",
9   "model_name": "string (e.g., llama3:8b)",
10  "model_version": "string (e.g., 8.0B)",
11  "weights_hash": "string (SHA-256 of weights)",
12  "model_source": "string (e.g., ollama-local)",
13  "inference_params": {
14    "temperature": "float",
15    "top_p": "float",
16    "top_k": "integer",
17    "max_tokens": "integer",
18    "seed": "integer|null",
19    "decoding_strategy": "string"
20  },
21  "params_hash": "string (SHA-256 of params)",
22  "environment": {
23    "os": "string",
24    "os_version": "string",
25    "architecture": "string",
26    "python_version": "string",
27    "hostname": "string",
28    "timestamp": "ISO 8601 datetime"
29  },
30  "environment_hash": "string (SHA-256)"
```

658

```

659 31 "code_commit": "string (git commit hash)",
660 32 "researcher_id": "string",
661 33 "affiliation": "string",
662 34 "timestamp_start": "ISO 8601 datetime",
663 35 "timestamp_end": "ISO 8601 datetime",
664 36 "output_text": "string (model output)",
665 37 "output_hash": "string (SHA-256 of output)",
666 38 "output_metrics": "object (task-specific)",
667 39 "execution_duration_ms": "float",
668 40 "logging_overhead_ms": "float",
669 41 "storage_kb": "float",
670 42 "system_logs": "string (raw system info)",
671 43 "errors": "array of strings"
672 44 }

```

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.

```

673 {
674   "prefix": {
675     "genai": "https://genai-prov.org/ns#",
676     "prov": "http://www.w3.org/ns/prov#"
677   },
678   "entity": {
679     "genai:prompt_c9644358": {
680       "prov:type": "genai:Prompt",
681       "genai:hash": "c9644358805b...",
682       "genai:task_category": "summarization"
683     },
684     "genai:model_llama3_8b": {
685       "prov:type": "genai:ModelVersion",
686       "genai:name": "llama3:8b",
687       "genai:source": "ollama-local"
688     },
689     "genai:output_590d0835": {
690       "prov:type": "genai:Output",
691       "genai:hash": "590d08359e7d..."
692     }
693   },
694   "activity": {
695     "genai:run_llama3_8b_sum_001_C1_rep0": {
696       "prov:type": "genai:RunGeneration",
697       "prov:startTime": "2026-02-07T21:54:34Z",
698       "prov:endTime": "2026-02-07T21:54:40Z"
699     }
700   },
701   "wasGeneratedBy": {
702     "_:wGB1": {
703
704
705

```

```

706      31      "prov:entity": "genai:output_590d0835",
707      32      "prov:activity": "genai:run_llama3_8b_..."
708      33    }
709      34  },
710      35  "used": {
711      36    "_:u1": {
712      37      "prov:activity": "genai:run_llama3_...",
713      38      "prov:entity": "genai:prompt_c9644358"
714      39    }
715      40  },
716      41  "agent": {
717      42    "genai:researcher_lucas_rover": {
718      43      "prov:type": "prov:Person",
719      44      "genai:affiliation": "UTFPR"
720      45    }
721      46  },
722      47  "wasAssociatedWith": {
723      48    "_:wAW1": {
724      49      "prov:activity": "genai:run_llama3_...",
725      50      "prov:agent": "genai:researcher_..."
726      51    }
727      52  }
728      53}

```

Received February 2026

729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752