Supplementary Material

Same Prompt, Different Answer: Exposing the Reproducibility Illusion in Large Language Model APIs

Lucas Rover   Yara de Souza Tadano
UTFPR – Universidade Tecnológica Federal do Paraná

February 2026

This document provides supplementary material for the main manuscript. Sections S1–S10 contain extended analyses, representative examples, and complete data tables referenced in the main text.

Contents

1 S1: Representative Output Examples

To illustrate *in practice* what deterministic and non-deterministic outputs look like, we present verbatim outputs from three models on the same input abstract (Vaswani et al., 2017 — "Attention Is All You Need") under identical greedy decoding conditions (t=0, seed = 42).

1.1 Deterministic Case: Gemma 2 9B (EMR = 1.000)

All five repetitions produced **character-for-character identical** output, confirmed by matching SHA-256 hashes.

Listing 1: Gemma 2 9B, extraction, C1, abstract 001 — all 5 repetitions produce this exact output.

```
{
  "objective": "To propose a new simple network
    architecture, the Transformer, based solely on
    attention mechanisms, dispensing with recurrence
    and convolutions entirely.",
  "method": "The Transformer architecture relies
    entirely on attention mechanisms...",
  "key_result": "BLEU score of 28.4 on the WMT 2014
    English-to-German translation task...",
  "model_or_system": "Transformer",
  "benchmark": "WMT 2014 English-to-German and
    English-to-French translation tasks"
}
```

Output hash (all 5 reps): `e7d7667b99bda09f...`
Verdict: Perfect bitwise reproducibility. Every character, whitespace, and newline is identical across runs.

1.2 Non-Deterministic Case: Claude Sonnet 4.5 (EMR = 0.020)

All five repetitions produced **distinct** outputs despite identical configuration (t=0, same prompt, same input):

Listing 2: Claude Sonnet 4.5, summarization, C1, abstract 001 — 5 distinct outputs from identical conditions.

```
Rep 0: "The paper introduces the Transformer, a novel
neural network architecture for sequence transduction
based entirely on attention mechanisms without
recurrence or convolutions..."

Rep 1: "This paper introduces the Transformer, a novel
network architecture for sequence transduction that
relies entirely on attention mechanisms without using
recurrence or convolutions..."

Rep 2: "The paper introduces the Transformer, a novel
sequence transduction architecture based entirely on
attention mechanisms without recurrence or
convolutions..."

```

```
Rep 3: "The paper introduces the Transformer, a novel
sequence transduction architecture based entirely on
attention mechanisms without recurrent or
convolutional layers..."

Rep 4: "The paper introduces the Transformer, a novel
neural network architecture for sequence transduction
based entirely on attention mechanisms without using
recurrence..."
```

Output hashes:

- Rep 0: `7032b33ee6ba79e5...`
- Rep 1: `6241ea4028d189e1...`
- Rep 2: `eaddc84b97889eec...`
- Rep 3: `dac6ac79c172e7ed...`
- Rep 4: `6a2b457ab6e9e279...`

Verdict: All five hashes are unique — zero exact matches from five attempts. The outputs are semantically equivalent (all correctly describe the Transformer architecture) but differ in phrasing: "neural network architecture" vs. "network architecture" vs. "sequence transduction architecture"; "without recurrence" vs. "without using recurrence" vs. "without recurrent layers." This exemplifies the core finding: API non-determinism manifests as *lexical variation with preserved semantics* — invisible without systematic logging, yet potentially impactful for automated pipelines, regression testing, and regulatory audit trails that expect exact reproducibility.

2 S2: Sources of Non-Determinism in Distributed Inference

Note on plausible mechanisms. The mechanisms discussed below are not directly observed within proprietary serving stacks. They are theoretically grounded factors consistent with known properties of distributed inference systems. We present them to motivate future investigation, not as confirmed explanations.

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.

(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 [Higham, 2002]. 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 [Micikevicius et al., 2018]. Yuan et al. [2025] 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 [Shoeybi et al., 2019], 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.

(4) Attention kernel non-determinism. FlashAttention [Dao et al., 2022], now the standard attention implementation in production LLM serving, introduces non-determinism through its parallelization strategy. Golden et al. [2024] 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.

(5) Dynamic batching and request scheduling. Production LLM serving systems use continuous batching [Yu et al., 2022, Kwon et al., 2023], where new requests are inserted into running batches at each decoding step. Different batch compositions lead to 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 [Leviathan et al., 2023] 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.

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 GGML Q4 quantization format, which uses integer arithmetic for the core computation.

Quasi-isolation probe: empirical evidence. 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—while major closed-source API models exhibit EMR ≤ 0.443. 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.

3 S3: Protocol Minimality — Ablation Analysis

To substantiate the protocol's minimality claim, we systematically removed each field group from the Run Card schema and assessed which audit questions became unanswerable. We decomposed the Run Card into eight finer-grained field groups: Identification, Model Context, Parameters, Input Content, Output Content, Hashing (all SHA-256 digests), Environment, and Overhead (timing and storage metadata).

We defined 10 audit questions that a reproducibility-oriented researcher might ask:

1. Can we verify the exact prompt used?
2. Can we confirm model identity?
3. Can we reconstruct inference parameters?
4. Can we verify input integrity?
5. Can we detect output tampering?
6. Can we compare outputs across runs?

7. Can we attribute environment effects?
8. Can we assess protocol overhead?
9. Can we trace full provenance?
10. Can we perform differential diagnosis?

Table 1: Ablation matrix: which audit questions (Q1–Q10) become unanswerable when each field group is removed.

Field Group	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Identification	×					×			×	
Model Context		×							×	×
Parameters			×						×	×
Input Content	×			×					×	
Output Content					×	×			×	
Hashing	×	×	×	×	×					×
Environment							×		×	×
Overhead								×		

× = question becomes unanswerable. The Hashing group affects 6/10 questions despite contributing only 410 bytes per run. Every row has at least one ×, confirming minimality.

Total overhead per run: ∼4,052 bytes. No field group is redundant.

4 S4: PROV Query Specifications and Results

We implemented three programmatic queries over the 4,104-run PROV dataset to demonstrate that PROV-based reasoning goes beyond what plain JSON logs provide.

Query 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.

Query 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.

Query 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." *Result:* Every output traces back to its complete generation context with no broken links—a guarantee that plain JSON logs cannot provide without custom graph-traversal code.

5 S5: Example PROV-JSON Document

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

Listing 3: Abbreviated PROV-JSON for a summarization run.

```
{
  "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_..."
    }
  },
  "used": {
    "_:u1": {
      "prov:activity": "genai:run_...",
      "prov:entity": "genai:prompt_c9644358"
    }
  },
  "agent": {
    "genai:researcher_lucas_rover": {
      "prov:type": "prov:Person",
      "genai:affiliation": "UTFPR"
    }
  },
  "wasAssociatedWith": {
    "_:wAW1": {
      "prov:activity": "genai:run_...",
      "prov:agent": "genai:researcher_..."
    }
  }
}
```

6 S6: JSON Extraction Quality

Two notable patterns emerge from the structured extraction task. First, LLaMA 3 never produces raw-valid JSON: all 570 extraction outputs contain preamble text before the JSON object, despite the prompt requesting "JSON only." After regex extraction, validity reaches 100% under greedy decoding (92.2% at t=0.7). GPT-4 always produces raw-valid JSON with 100% schema compliance.

Second, within-abstract field-level exact match rates confirm the reproducibility hierarchy. Under greedy decoding, LLaMA 3 achieves field EMR 0.982–0.989 overall, while GPT-4 shows 0.757–0.767, with open-ended fields (`method`: 0.667) lagging behind structured fields (`benchmark`: 0.863). At t=0.7, this gap widens to 4–5×.

Note: The complete JSON quality metrics table (`table_json_metrics.tex`) is available in the project repository.

7 S7: Prompt Card Example

Complete Prompt Card for the summarization task:

Listing 4: 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 covering:
    (1) main contribution, (2) methodology,
    (3) key result.",
  "assumptions": [
    "Input is a single English scientific abstract",
    "Abstract contains identifiable methodology
     and quantitative results"
  ],
  "limitations": [
    "Open-ended phrasing allows high output variance",
    "No explicit output-format constraint"
  ],
  "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"}
  ]
}
```

8 S8: Representative Prompt Templates

The exact prompt templates used for each experimental task ({`abstract`} is replaced 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]\n Turn 2: Now revise the summary to be more specific about
the quantitative results.\n Turn 3: Add one sentence about the limitations or future
work mentioned.
```

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:
```

9 S9: Experimental Coverage Matrix

Table 2: Number of abstracts (runs) per model–task–condition. Dash = not evaluated.

Model	**Task**	**C1**	**C2**	**C3 (t=0)**	**C3 (t=0.3)**	**C3 (t=0.7)**
LLaMA 3 8B	Extr.	30 (150)	30 (150)	30 (90)	30 (90)	30 (90)
	Summ.	30 (150)	30 (150)	30 (90)	30 (90)	30 (90)
	Multi	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Mistral 7B	Extr.	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Summ.	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Gemma 2 9B	Extr.	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Summ.	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
GPT-4	Extr.	–	30 (150)	17 (51)	17 (51)	14 (42)
	Summ.	3 (8)*	30 (150)	30 (90)	30 (90)	30 (90)
Claude 4.5	Extr.	10 (49)†	10 (50)	10 (30)	10 (30)	10 (30)
	Summ.	10 (50)	10 (50)	10 (30)	10 (30)	10 (30)
	Multi	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
Gemini 2.5 Pro	Multi	10 (50)	–	–	–	–
	RAG	10 (50)	–	–	–	–
DeepSeek Chat	Extr./Summ.	10 (100)	–	–	–	–
Perplexity Sonar	Extr./Summ.	10 (100)	–	–	–	–
Together AI LLaMA	Extr./Summ.	10 (100)	10 (100)	–	–	–

*GPT-4 C1 summarization: only 3 abstracts before quota exhaustion. †Claude C1 extraction: 49 runs (1 API timeout). An additional 200 chat-format control runs (LLaMA 3) are not shown.

10 S10: Environment and Provenance Transparency

Execution Environment

Table 3: Execution environment for local model inference.

Component	Value
CPU	Apple M4 (10-core)
Unified memory	24 GB
Operating system	macOS 14.6 (Darwin 24.6.0)
Architecture	arm64
Python	3.14.3 (Clang 17.0.0)
Ollama	v0.15.5
Quantization	Q4_0 (GGML) for all local models
Inference mode	Single-request (no concurrent batching)

All local runs share the same environment hash (`cd45d428...`), confirming no system changes during the experiment window.

Code Versioning Timeline

Table 4: Experiment phases and git status.

Date	Phase	code_commit	Runs
Feb 7, 2026	Initial experiments	`no-git-repo`	330
Feb 7, 2026	Repository initialized	`ff025ef`	—
Feb 8–9	Expanded experiments	`957a604–0cba4f1`	1,874
Feb 9	Claude experiments	`e57ae19–df2b07e`	1,500
Feb 10	Gemini experiments	`b2e9440`	200
Feb 9	Chat-format control	`0cba4f1`	200
Total with valid code_commit			3,774 (92%)
Total without code_commit			330 (8%)

Post-Hoc Integrity Verification for Pre-Git Runs

For the 330 pre-git runs, we verified: (1) prompt hashes match the committed templates; (2) output hashes remain valid SHA-256 digests; (3) environment hashes are identical to post-git runs on the same machine.

Weights Hash Coverage

Table 5: Weights hash field population across all 4,104 run records.

Model	**weights_hash**	**Runs**	**Explanation**
LLaMA 3 8B	(empty)	1,340	Pre-API integration
LLaMA 3 8B	`365c0bd3...`	100	Multi-turn/RAG
Mistral 7B	(empty)	380	Pre-API integration
Mistral 7B	`6577803a...`	100	Multi-turn/RAG
Gemma 2 9B	`ff02c370...`	480	All runs
GPT-4	proprietary-not-available	724	API model
Claude Sonnet 4.5	proprietary-not-available	480	API model
Gemini 2.5 Pro	proprietary-not-available	100	API model
DeepSeek Chat	proprietary-not-available	100	API model
Perplexity Sonar	proprietary-not-available	100	API model
Together AI LLaMA	open-weight-cloud-served	200	Cloud-served
Total populated		**2,384**	(58%)
Total empty		**1,720**	(42%)

The 1,720 empty entries are exclusively from LLaMA 3 and Mistral 7B single-turn experiments conducted before the Ollama weights-hash API was integrated into the protocol.

Representative Run Record

Listing 5: Abbreviated run record (LLaMA 3 8B, extraction, C1, rep 0).

```
{
  "run_id": "llama3_8b_extraction_abs_001_C1_...",
  "task_category": "structured_extraction",
  "prompt_hash": "ddc1746ca23ddafc...",
  "input_hash": "4e873413a989a6fe...",
  "model_name": "llama3:8b",
  "model_version": "8.0B",
  "weights_hash": "<SHA-256 from Ollama API>",
  "inference_params": {
    "temperature": 0.0, "seed": 42,
    "decoding_strategy": "greedy"
  },
  "params_hash": "5f7d7b01f15d5632...",
  "environment_hash": "cd45d42899c58da3...",
  "output_hash": "d89180f3e1a11858...",
  "execution_duration_ms": 7816.7,
  "logging_overhead_ms": 34.2,
  "storage_kb": 4.23
}
```

References

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information*

Processing Systems, volume 35, 2022.

Alicia Golden, Samuel Hsia, Fei Sun, Bilge Acun, Basil Hosmer, Yejin Lee, Zachary DeVito, Jeff Johnson, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Is flash attention stable? *arXiv preprint arXiv:2405.02803*, 2024.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, 2nd edition, 2002.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles*, 2023.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 2023.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation*, pages 521–538. USENIX Association, 2022.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems*, volume 38. Curran Associates, Inc., 2025. arXiv preprint.