Supplementary Information

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

Lucas Rover, Eduardo Tadeu Bacalhau, Anibal Tavares de Azevedo & Yara de Souza Tadano
UTFPR – Universidade Tecnológica Federal do Paraná

This document provides Supplementary Information for the main manuscript. Sections S1–S10 contain full prompt templates, input descriptions, protocol analyses, statistical details, and experimental coverage data referenced in the main text.

Contents

S1 Full Prompt Templates

The exact prompt templates used for each of the four experimental tasks are reproduced below. In all templates, {`abstract`} is replaced at runtime with the full text of the target scientific abstract, and {`retrieved_passage`} is replaced with the domain-relevant context passage for RAG tasks.

S1.1 Task 1: Scientific Summarization

Listing 1: Prompt template for 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.

Abstract: {abstract}

Summary:
```

This is an open-ended generation task. The model must produce three sentences of plain text with no structured markup. The unconstrained output format permits substantial lexical variation across runs.

S1.2 Task 2: Structured Extraction

Listing 2: Prompt template for Task 2 (structured extraction).

```
Extract the following fields from the scientific abstract
below. Return JSON only, no explanation.

Fields: objective, method, key_result, model_or_system,
benchmark

Abstract: {abstract}

JSON:
```

This is a constrained generation task. The model must return a JSON object with exactly five fields. The structured output format reduces but does not eliminate lexical variation.

S1.3 Task 3: Multi-Turn Refinement (3 turns)

Listing 3: Prompt template for Task 3 (multi-turn refinement, 3 turns).

```
Turn 1:
Summarize the following scientific abstract in exactly
3 sentences. Cover: (1) the main contribution,
(2) the methodology used, and (3) the key quantitative
result.

Abstract: {abstract}

Summary:
```

```

Turn 2:
Now revise the summary to be more specific about
the quantitative results.

Turn 3:
Add one sentence about the limitations or future work
mentioned.
```

Each turn is submitted as a separate message in the conversation history. For local models, Ollama's `/api/chat` endpoint is used with an accumulated `messages` array. For API models, the provider's native chat interface is used with the same message structure. The full conversation state (including all prior turns) is hashed and logged as `conversation_history_hash` in the Run Card.

S1.4 Task 4: RAG Extraction

Listing 4: Prompt template for Task 4 (RAG extraction).

```
Using the context passage below and the scientific
abstract, extract the following fields. Return JSON only.

Context: {retrieved_passage}
Abstract: {abstract}

Fields: objective, method, key_result, model_or_system,
benchmark

JSON:
```

This task augments the structured extraction prompt (Task 2) with a retrieved context passage prepended to the abstract. The context passage is a domain-relevant text segment that provides additional information about the paper's topic. The purpose is to test whether augmenting the prompt with external context affects reproducibility.

S2 Input Abstracts

The experiment corpus comprises 30 scientific abstracts drawn from highly cited publications in machine learning, computer vision, natural language processing, and reinforcement learning. Table S1 lists all abstracts by number, authors, year, title, and venue. Full text and DOIs are available in the repository at `data/inputs/abstracts.json`.

Table S1: Input abstracts used in all experiments (30 total).

#	Citation
1	Vaswani et al. (2017) – Attention Is All You Need, NeurIPS
2	Devlin et al. (2019) – BERT: Pre-training of Deep Bidirectional Transformers, NAACL
3	Brown et al. (2020) – Language Models are Few-Shot Learners, NeurIPS
4	Raffel et al. (2020) – Exploring the Limits of Transfer Learning with T5, JMLR
5	Wei et al. (2022) – Chain-of-Thought Prompting Elicits Reasoning in LLMs, NeurIPS
6	Goodfellow et al. (2014) – Generative Adversarial Nets, NeurIPS

#	Citation
7	He et al. (2016) – Deep Residual Learning for Image Recognition, CVPR
8	Kingma & Welling (2014) – Auto-Encoding Variational Bayes, ICLR
9	Hochreiter & Schmidhuber (1997) – Long Short-Term Memory, Neural Computation
10	Radford et al. (2021) – Learning Transferable Visual Models (CLIP), ICML
11	Ramesh et al. (2022) – Hierarchical Text-Conditional Image Generation (DALL-E 2), arXiv
12	Rombach et al. (2022) – High-Resolution Image Synthesis with Latent Diffusion Models, CVPR
13	Touvron et al. (2023) – LLaMA: Open and Efficient Foundation Language Models, arXiv
14	Ouyang et al. (2022) – Training Language Models to Follow Instructions (InstructGPT), NeurIPS
15	OpenAI (2023) – GPT-4 Technical Report, arXiv
16	Chowdhery et al. (2022) – PaLM: Scaling Language Modeling with Pathways, JMLR
17	Liu et al. (2019) – RoBERTa: A Robustly Optimized BERT Pretraining Approach, arXiv
18	Lewis et al. (2020) – BART: Denoising Sequence-to-Sequence Pre-training, ACL
19	Dosovitskiy et al. (2021) – An Image is Worth 16x16 Words (ViT), ICLR
20	Kirillov et al. (2023) – Segment Anything, ICCV
21	Ho et al. (2020) – Denoising Diffusion Probabilistic Models, NeurIPS
22	Schulman et al. (2017) – Proximal Policy Optimization Algorithms, arXiv
23	Silver et al. (2016) – Mastering the Game of Go (AlphaGo), Nature
24	Kipf & Welling (2017) – Semi-Supervised Classification with GCNs, ICLR
25	Mikolov et al. (2013) – Efficient Estimation of Word Representations (Word2Vec), ICLR Workshop
26	Bahdanau et al. (2015) – Neural Machine Translation by Jointly Learning to Align and Translate, ICLR
27	Rezende & Mohamed (2015) – Variational Inference with Normalizing Flows, ICML
28	Zoph & Le (2017) – Neural Architecture Search with Reinforcement Learning, ICLR
29	Hu et al. (2022) – LoRA: Low-Rank Adaptation of Large Language Models, ICLR
30	Touvron et al. (2023) – Llama 2: Open Foundation and Fine-Tuned Chat Models, arXiv

The corpus spans publication years from 1997 to 2023 and includes foundational works (Transformers, GANs, LSTM), scaling studies (GPT-3, PaLM, T5), alignment methods (InstructGPT, RLHF), efficient adaptation (LoRA), and multimodal models (CLIP, DALL-E 2, SAM). This diversity ensures that our findings are not artefacts of a single subfield or writing style.

S3 Retrieved Contexts for RAG

For Task 4 (RAG Extraction), each of the 30 input abstracts is augmented with a domain-relevant context passage prepended to the prompt. These passages serve as simulated retrieval results, providing additional topical information that the model can use when extracting structured fields.

The context passages were curated to satisfy two criteria: (1) domain relevance to the target abstract's topic (e.g., a passage about attention mechanisms for the Transformer abstract), and (2) non-overlap with the abstract text itself, to test whether the model integrates external context consistently across repeated runs.

Each context passage is approximately 150–300 words in length and is drawn from survey papers, textbook descriptions, or related work sections of the cited papers. The exact text of all 30 context passages, along with SHA-256 hashes verifying their integrity, is available in the project repository at `data/inputs/rag_contexts.json`.

The key finding for RAG tasks is that context augmentation does not substantially improve API reproducibility: GPT-4 achieves EMR = 0.230 on RAG extraction (compared to 0.443 on standard extraction under C1), and Gemini 2.5 Pro achieves EMR = 0.070. Local models maintain near-perfect reproducibility regardless of context augmentation (see main text, Table 5).

S4 API Payload Documentation

To address potential "apples-to-oranges" concerns regarding differences in API request structures across providers, we document the exact JSON payload structures sent to each of the seven inference endpoints. All payloads were constructed deterministically and logged as part of the Run Card.

S4.1 Ollama (Local Models)

Single-turn tasks (Tasks 1–2) use `POST /api/generate`:

Listing 5: 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 an accumulated `messages` array containing alternating `user` and `assistant` roles. No system prompt, stop sequences, or post-processing are applied.

S4.2 OpenAI (GPT-4)

Accessed via the `openai` Python SDK v1.59.9:

Listing 6: OpenAI Chat Completions payload.

```
{
  "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. OpenAI documents that the `seed` parameter provides "mostly deterministic" outputs but does not guarantee bitwise reproducibility.

S4.3 Anthropic (Claude Sonnet 4.5)

Accessed via `urllib` (no SDK dependency):

Listing 7: Anthropic Messages API payload.

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

Key difference: The Anthropic API does not support a `seed` parameter. The seed value in the Run Card is marked `seed_status: "logged-only-not-sent-to-api"`. No system message or stop sequences were used.

S4.4 Google (Gemini 2.5 Pro)

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

Listing 8: Google Gemini API payload.

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

Key difference: 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 consume the `maxOutputTokens` budget, hence the higher limit (8,192 vs. 1,024 for other models). Multi-turn tasks use the same endpoint with an accumulated `contents` array alternating `role: "user"` and `role: "model"`.

S4.5 DeepSeek (DeepSeek Chat)

Accessed via the OpenAI-compatible API:

Listing 9: DeepSeek Chat API payload.

```
{
  "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.

S4.6 Perplexity (Perplexity Sonar)

Accessed via the Perplexity API:

Listing 10: Perplexity Sonar API payload.

```
{
  "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).

S4.7 Together AI (LLaMA 3 8B)

Accessed via the Together AI OpenAI-compatible API:

Listing 11: Together AI API payload.

```
{
  "model": "meta-llama/Meta-Llama-3-8B-Instruct",
  "messages": [
    {"role": "user", "content": "<prompt>"}
  ],
  "temperature": 0.0,
  "seed": 42,
  "max_tokens": 1024
}
```

Together AI serves the same open-weight LLaMA 3 8B model used locally, enabling a quasi-isolation probe: differences in EMR between local and cloud-served deployments of the same weights can be attributed to infrastructure effects (tensor parallelism, dynamic batching) rather than model architecture or training differences.

S4.8 Key Symmetry Points

Across all nine model deployments: (1) identical prompt text (verified by `prompt_hash`); (2) identical temperature ($t = 0.0$); (3) identical maximum 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 normalisation of outputs.

S5 Protocol Comparison Table

Table S2 provides a systematic feature-by-feature comparison of our protocol with five existing experiment-tracking and evaluation tools. The key distinction is not merely one of tooling but of scientific capability: existing tools log what happened during training (parameters, metrics, artefacts), whereas our protocol enables answering questions about whether two generative outputs are provably derived from identical configurations, which exact factor caused a divergence, and whether an output has been tampered with post-generation.

Table S2: Comparison of our protocol with existing reproducibility tools and frameworks. Checkmarks indicate full support; tildes (∼) indicate partial support; dashes (–) indicate no support.

Feature	**Ours**	**MLflow**	**W&B**	**DVC**	**OAI Evals**	**LangSmith**
Prompt versioning	✓	–	∼	–	∼	∼
Output hashing	✓	–	–	✓	–	–
PROV integration	✓	–	–	–	–	–
Env. fingerprinting	✓	∼	∼	∼	–	–
Overhead <1%	✓	∼	∼	N/A	N/A	∼
Inference-specific design	✓	–	–	–	✓	✓
Open standard	✓	–	–	–	–	–
Seed & param logging	✓	✓	✓	–	✓	✓
Model weights hashing	✓	–	∼	✓	–	–
Local-first (no cloud)	✓	✓	–	✓	–	–

OAI Evals = OpenAI Evals; W&B = Weights & Biases. Assessment based on publicly documented features as of February 2026.

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

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

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

OpenAI Evals 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 offers tracing and evaluation for LLM applications. It captures detailed execution traces but uses a proprietary format and requires cloud connectivity.

S6 Protocol Minimality – Ablation Study

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. The analysis was conducted over 1,864 run records.

S6.1 Field Groups and Audit Questions

We decomposed the Run Card into eight field groups:

1. **Identification**: `run_id`, `task_category`, `task_id`, `interaction_regime`
2. **Model Context**: `model_name`, `model_version`, `model_source`, `weights_hash`
3. **Parameters**: `inference_params`, `params_hash`
4. **Input/Output**: `prompt_text`, `input_text`, `input_hash`, `output_text`, `output_hash`, `output_metrics`
5. **Hashing**: `prompt_hash`, `input_hash`, `output_hash`, `params_hash`, `environment_hash`
6. **Environment**: `environment`, `environment_hash`, `code_commit`
7. **Timing**: `timestamp_start`, `timestamp_end`, `execution_duration_ms`, `logging_overhead_ms`
8. **Overhead**: `logging_overhead_ms`, `storage_kb`

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

- Q1. Can we verify the exact prompt used?
- Q2. Can we verify the model identity?
- Q3. Can we verify the generation parameters?
- Q4. Can we detect output tampering?
- Q5. Can we verify the execution environment?
- Q6. Can we reproduce the exact timing?
- Q7. Can we assess protocol overhead?
- Q8. Can we trace full provenance?
- Q9. Can we verify input integrity?
- Q10. Can we compare outputs across runs?

S6.2 Ablation Matrix

Table S3 shows the ablation results. Each × indicates that removing the corresponding field group renders the audit question unanswerable.

Table S3: Ablation matrix: which audit questions (Q1–Q10) become unanswerable when each field group is removed. × = question becomes unanswerable.

Field Group	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	**Lost**
Identification								×		×	2
Model Context		×								×	2
Parameters			×							×	2
Input/Output	×			×					×	×	4
Hashing	×		×	×	×				×	×	6
Environment					×						1
Timing						×	×				2
Overhead							×				1

The Hashing group affects 6/10 questions despite contributing only ∼410 bytes per run (<10% of the record). Every row has at least one ×, confirming minimality: no field group can be removed without losing audit capability.

S6.3 Storage and Overhead Analysis

Table S4 reports the storage cost and computational overhead of each field group.

Table S4: Storage and overhead per field group, measured across 1,864 run records.

Field Group	**Mean bytes**	**% of record**	**Est. ms**	**Verdict**
Identification	158.6	3.8%	0.0	Essential (low)
Model Context	106.1	2.6%	0.0	Essential (low)
Parameters	206.5	5.0%	1.3	Essential (low)
Input/Output	2,413.8	58.6%	0.0	Essential (med)
Hashing	409.0	9.9%	10.2	Essential (high)
Environment	547.9	13.3%	8.9	Essential (low)
Timing	163.8	4.0%	2.5	Essential (low)
Overhead	45.8	1.1%	2.5	Essential (low)
Total	∼4,052	100%	∼25	

Total overhead per run: approximately 4,052 bytes storage and 25 ms computation, representing <1% of typical inference time.

S7 Chat-Format Control Experiment

To assess whether the prompt-format difference between completion-style and chat-style API calls contributes to the observed reproducibility gap between local and API models, we conducted a control experiment running LLaMA 3 8B through two different Ollama endpoints.

S7.1 Design

- **Completion format**: `POST /api/generate` (raw prompt string)
- **Chat format**: `POST /api/chat` (structured messages with `role: "user"`)
- **Scale**: 10 abstracts × 2 tasks (summarisation, extraction) × 2 conditions (C1 fixed seed, C2 variable seed) × 5 repetitions = 200 runs per format, 400 runs total
- **Decoding**: Greedy ($t = 0$, seed = 42 for C1)

S7.2 Results

Table S5 presents the per-format metrics. The two formats produce identical aggregate metrics for both tasks.

Table S5: Chat-format control experiment: LLaMA 3 8B, completion vs. chat format (200 runs each).

Task	**Format**	**EMR**	**NED**	**ROUGE-L**
Summarisation	Completion	0.929	0.007	0.992
Summarisation	Chat	0.929	0.007	0.992
Extraction	Completion	1.000	0.000	1.000
Extraction	Chat	1.000	0.000	1.000

EMR = Exact Match Rate (fraction of abstract groups where all 5 repetitions are identical); NED = Normalised Edit Distance; ROUGE-L = longest common subsequence F-measure.

Per-abstract EMR values are also identical across formats for all 10 abstracts in both tasks. For summarisation, abstracts 1 and 3 show EMR = 0.644 in both formats (indicating some within-group variation attributable to the warm-up effect on the first repetition), while all other abstracts achieve EMR = 1.000 in both formats. For extraction, all 10 abstracts achieve EMR = 1.000 in both formats.

S7.3 Interpretation

The chat-format control provides evidence that prompt formatting is **not** a source of the reproducibility gap between local and API models. The identical metrics across completion and chat formats indicate that the local model produces the same outputs regardless of whether the prompt is presented as a raw string or as a structured chat message. This result supports the interpretation that the local–API gap is driven by infrastructure-level factors (tensor parallelism, dynamic batching, mixed-precision non-determinism) rather than by differences in prompt formatting conventions.

S8 Reproducibility Checklist

The following 15-item checklist is designed for self-assessment of reproducibility in generative AI studies. Each item maps to a specific field or artefact in our protocol. We organise the checklist into four categories.

Category 1: Prompt Documentation (4 items)

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-turn/multi-turn)? [`Prompt Card: interaction_regime`]

Category 2: Model and Environment (4 items)

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`]

Category 3: Execution and Output (5 items)

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`]

Category 4: Provenance (2 items)

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

A study satisfying all 15 items achieves the highest level of reproducibility documentation under our framework. We recommend that studies report at minimum items 1, 5, 9, 10, and 11 (see main text for discussion).

S9 Statistical Test Results

S9.1 Methodology

We applied the Holm–Bonferroni sequential rejection procedure to control the family-wise error rate at $\alpha = 0.05$ across all pairwise comparisons. The test battery comprised 68 individual tests:

- 24 Fisher's exact tests (3 local models $\times$ 4 API models $\times$ 2 tasks): testing whether the proportion of abstracts with perfect reproducibility (EMR = 1.0) differs between local and API models.
- 24 Mann–Whitney U tests (same pairings): testing whether per-abstract EMR distributions differ between local and API models.
- 18 Wilcoxon signed-rank tests (3 local $\times$ 4 API $\times$ 2 tasks, where paired data available on shared abstracts): testing within-abstract EMR differences.
- 2 aggregate Mann–Whitney U tests (all local vs. all API, per task): testing the overall local–API gap.

S9.2 Holm–Bonferroni Procedure

The Holm–Bonferroni method orders all $m = 68$ p-values from smallest to largest and rejects hypothesis $H_{(k)}$ if $p_{(k)} \leq \alpha/(m - k + 1)$ for all $k' \leq k$. This provides strong control of the family-wise error rate without assuming independence among tests.

S9.3 Summary Results

Of the 68 tests, **51 were rejected** (significant) and 17 were not rejected after correction:

- All 24 Fisher's exact tests involving LLaMA 3 8B, Gemma 2 9B, and Mistral 7B versus GPT-4, Claude, and Perplexity were significant.
- All aggregate local-vs.-API tests were significant ($p_{\text{adj}} < 10^{-9}$).
- The 17 non-rejected tests primarily involved comparisons with DeepSeek Chat, which has the highest API reproducibility (EMR = 0.800 for extraction), making the gap with some local models non-significant at the corrected threshold.

S9.4 Representative Results

Table S6 presents a selection of representative Holm–Bonferroni-corrected results spanning the range from the most significant to the borderline cases.

Table S6: Representative Holm–Bonferroni-corrected test results (selection of 16 from 68 total). Rank = position in the sorted p-value list.

Test	**Rank**	$p_{\mathbf{orig}}$	$p_{\mathbf{adj}}$	**Reject**
Fisher: LLaMA vs. GPT-4, Extr.	2	<0.001	<0.001	Yes
Fisher: LLaMA vs. Perplexity, Extr.	3	<0.001	<0.001	Yes
Fisher: Gemma vs. GPT-4, Summ.	4	<0.001	<0.001	Yes
Fisher: LLaMA vs. GPT-4, Summ.	5	<0.001	<0.001	Yes
Aggregate: Local vs. API, Summ.	6	9.3×10^{-17}	5.9×10^{-15}	Yes
Aggregate: Local vs. API, Extr.	7	1.0×10^{-14}	6.4×10^{-13}	Yes
MW: LLaMA vs. GPT-4, Summ.	8	4.2×10^{-11}	2.6×10^{-9}	Yes
Fisher: Gemma vs. Claude, Extr.	33	0.00012	0.0043	Yes
Fisher: Mistral vs. Claude, Extr.	47	0.0011	0.024	Yes
Wilcoxon: Mistral vs. Claude, Summ.	48	0.0020	0.041	Yes
Fisher: LLaMA vs. DeepSeek, Extr.	50	0.0020	0.039	Yes
MW: Gemma vs. DeepSeek, Summ.	51	0.0025	0.045	Yes
Fisher: LLaMA vs. DeepSeek, Summ.	53	0.0074	0.118	No
Fisher: Gemma vs. DeepSeek, Extr.	61	0.033	0.260	No
MW: Mistral vs. DeepSeek, Extr.	62	0.054	0.381	No
Fisher: Mistral vs. DeepSeek, Summ.	68	0.656	0.656	No

MW = Mann–Whitney U test. All p-values are two-sided. The complete set of 68 tests is available in the repository at `analysis/enhanced_statistical_results.json`.

S9.5 Effect Sizes

Cohen's h effect sizes for all pairwise Fisher comparisons ranged from 0.40 (Mistral vs. DeepSeek, summarisation – medium effect) to 3.14 (Gemma vs. Claude/Perplexity, summarisation – very large effect). All comparisons involving Gemma 2 9B vs. any API model yielded very large effect sizes ($h > 1.5$), reflecting Gemma's perfect reproducibility (EMR = 1.000) against API models' substantially lower rates.

S10 Experimental Coverage Matrix

Table S7 reports the number of abstracts and total runs per model–task–condition combination. The study encompasses 3,904 experimental runs across 9 model deployments, 4 tasks, and up to 5 experimental conditions.

Table S7: 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 summarisation: only 3 abstracts (8 runs) before API quota exhaustion. †Claude C1 extraction: 49 runs (1 API timeout returned empty output). An additional 200 chat-format control runs (LLaMA 3 via `/api/chat`) are documented in Section S7 but not shown here.

S10.1 Experimental Conditions

The five experimental conditions are:

C1 (Fixed seed): $t = 0$, seed = 42, 5 repetitions per abstract. Tests baseline reproducibility under maximal determinism settings.

C2 (Variable seed): $t = 0$, seeds $\in \{42, 123, 456, 789, 1024\}$, 5 repetitions per abstract. Tests whether changing the seed affects output under greedy decoding.

C3 ($t = 0.0$): Explicit temperature 0 with 3 repetitions using different seeds. Overlaps with C1/C2 to verify consistency.

C3 ($t = 0.3$): Low temperature with 3 repetitions. Tests near-greedy regime.

C3 ($t = 0.7$): Moderate temperature with 3 repetitions. Tests effect of increased stochasticity.

S10.2 Total Runs Summary

Table S8: Total runs by model category.

Category	Runs
Local models (LLaMA, Mistral, Gemma)	2,400
API models (GPT-4, Claude, Gemini, DeepSeek, Perplexity)	1,304
Cloud-served open-weight (Together AI)	200
Grand total	**3,904**

Excludes 200 chat-format control runs (see Section S7). Including controls, the total is 4,104 runs.