Single-Prompt Evaluation Can Mislead: Auditing Structured Compliance under Benign Prompt Drift

Anonymous Author(s)

Abstract

Single-prompt evaluation is often treated as a stable proxy for instruction-following and structured-output reliability. We present **Prompt Drift Lab**, an *audit-style* evaluation that holds tasks and decoding settings fixed while applying (i) benign prompt variants (rephrases and constraint framing) and (ii) an implicit vs. explicit instruction condition. We evaluate three generation models (ChatGPT, Claude, Gemini) on two short tasks (Q3–Q4) with four prompt variants (baseline/weak/long/conflict) and two instruction triggers (explicit/implicit), yielding 16 outputs per model. Each output is scored under a **judge-contract** by two out-of-family judges (cross-model judging), with self-judging as a supporting check. Across the same tasks, harmless prompt changes shift mean conclusions by multiple points, while instruction explicitness is an even stronger lever (e.g., Gemini averages 9.31 under explicit triggers but 0.50 under implicit triggers). We also treat **invalid** evaluations (schema violations / non-compliant judging) as first-class artifacts and bucket them into a lightweight failure taxonomy. All prompts, protocols, raw outputs, judged records, and summary tables are versioned and included as auditable artifacts.

1 Introduction: an audit stance

Problem. In practice, many evaluations rely on a single prompt snapshot. When a model "fails" or "passes" under that snapshot, the conclusion is often reported as if stable. However, prompt wording is a part of the evaluation protocol—and minor rephrases can change model behavior.

This paper's stance. We *audit* evaluation stability. We do not propose a new prompting recipe. Instead, we ask:

> *If we keep tasks and decoding fixed, how often do benign prompt variants and instruction explicitness flip conclusions about structured instruction-following?*

Audit artifacts. The contribution is evidence that can be inspected and reproduced: prompts, protocol, raw outputs, judged records, invalid logs, and scripts that regenerate all tables/figures.

Table 1: Inter-judge agreement (main method). Lower MAE and higher exact matches indicate closer scoring.

pair	n	MAE(total)	exact
chatgpt vs claude	16	1.06	7
chatgpt vs gemini	16	0.12	14
claude vs gemini	16	1.25	6

2 Setup and judge-contract

Tasks and variants. The audit fixes two tasks (Q3–Q4) and applies four prompt variants: *baseline*, *weak* (reduced constraint strength), *long* (more verbose framing), and *conflict* (deliberately conflicting phrasing). Each run also includes an *explicit* trigger (directly states the required structured format) and an *implicit* trigger (weaker/indirect instruction).

Judge-contract. Judges must (i) follow a fixed rubric, (ii) produce per-dimension scores A–E, and (iii) provide short evidence snippets. The contract is designed to make *violations visible*. Common violation modes include:

- **Schema/format break:** missing required fields or structure.
- **Instruction drift:** ignoring the explicit format requirement.
- **Meta-judging:** discussing the rubric without scoring (evaluation pollution).

Cross-model judging. Each generator model is evaluated by two other models (out-of-family judges) to reduce in-family bias; we also report self-judging as a supporting signal.

3 Findings: stability failures under prompt drift

F1: explicitness dominates stability. Figure 1 shows a large explicit–implicit gap across prompt variants (main method). In summary aggregates (Table 2), Gemini averages 9.31 under explicit triggers but 0.50 under implicit triggers; Claude collapses to 0.00 under implicit triggers. This indicates that "implicit" compliance is a distinct and often brittle capability under a structured-output contract.

F2: benign variants change conclusions without changing tasks. Tables 3–4 break down mean scores by question and prompt variant. For ChatGPT, moving

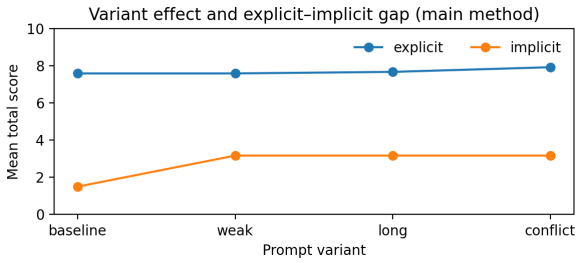

Figure 1: Variant effect and explicit–implicit gap (main method; mean total score).

Table 2: Main summary (cross-model judging; mean total and explicit/implicit means).

model	avg	exp	imp	zeros	perfect	n
chatgpt	8.56	9.38	7.75	1	3	16
claude	2.19	4.38	0.00	12	0	16
gemini	4.91	9.31	0.50	7	0	16

from baseline to conflict increases the average across Q3–Q4 from 6.13 to 9.63 (+3.50). For Gemini, the same change decreases the average from 5.38 to 4.75 (-0.63). These shifts occur while holding tasks and decoding fixed.

F3: single-prompt evaluation can mislead. A single prompt snapshot can select for a "good" or "bad" framing and thereby overstate (or understate) reliability. The audit artifacts make these flips traceable to concrete outputs and judged records rather than post-hoc narrative.

4 Invalid outputs as audit evidence

Why invalid logs matter. In addition to low scores, we observe cases where a judge fails to follow the judge-contract (e.g., missing required fields or refusing the schema). If a study silently drops such cases, it can bias the reported average and hide protocol brittleness.

Case artifact (from logs). A common pattern is a judge returning free-form commentary without the required A–E score fields. Under a contract, this is not "missing data"—it is a visible protocol failure that should be counted and reported.

> *"I cannot score this reliably from the provided answer. Here is a general discussion of the rubric..."*

Taxonomy. We bucket invalid cases into four primary categories: (A) schema/format, (B) instruction deviation, (D) robustness/instability, and (E) evaluation pollution (rubric-gaming / meta-judging). The buckets are generated by a transparent rule-based script over judge notes/evidence (included in the supplement), making the mapping auditable and modifiable.

Interpretation for workshop readers. Invalid logs are not "noise" to discard: they are evidence about the brittleness of the evaluation protocol itself. Reporting an average without the invalid rate can hide the true cost of enforcing structure.

Table 3: Main method: mean score for Q3 by prompt variant.

generator	baseline	weak	long	conflict
chatgpt	7.50	9.50	9.50	9.75
claude	4.25	4.50	4.25	4.50
gemini	4.00	4.75	4.75	4.75

Table 4: Main method: mean score for Q4 by prompt variant.

generator	baseline	weak	long	conflict
chatgpt	4.75	8.75	9.25	9.50
claude	0.00	0.00	0.00	0.00
gemini	6.75	4.75	4.75	4.75

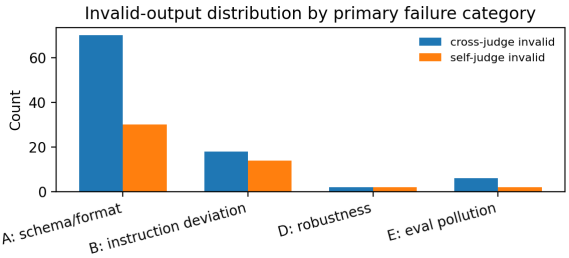

Figure 2: Invalid-output distribution by primary failure category (from invalid evaluation logs).

5 Discussion, limitations, and what to do with this

Scope. This audit is intentionally small (2 tasks, 4 variants, 3 generators) so that every artifact remains inspectable. We do not claim these numbers generalize to all tasks or settings.

Interpretation. The key takeaway is not "model X is better" but "evaluation conclusions can flip under protocol-near changes". If a benchmark relies on single-prompt evaluation, reporting should include a *prompt sensitivity check* or explicitly state that results are prompt-conditioned.

Practical reporting recommendations. For workshop readers designing evaluations, we recommend reporting:

- **Prompt sensitivity:** results across a small set of benign rephrases, not a single snapshot.
- **Invalid rate:** how often enforcing the schema/judge-contract produces non-usable evaluations.
- **Artifact pointers:** a stable mapping from each reported number to raw outputs and judged records.

Limitations. The tasks are short and the rubric is tailored to structured outputs; broader task families may require different contracts. Cross-model judging reduces in-family bias but does not eliminate judge errors; we therefore surface invalid logs explicitly rather than smoothing them away.

6 Related work (brief)

This paper is closest in spirit to work that treats prompting and evaluation protocols as first-class objects. We use prompting variants as benign perturbations rather than as an optimization method (see surveys of prompting methods such as [2]). Our judging setup follows the LLM-as-a-judge framing but emphasizes contract violations and invalid logs as auditable outputs rather than treating them as noise (cf. [3]).

7 Reproducibility

All prompts, protocols, raw outputs, judged records, and summary tables are versioned. Paper figures/tables are generated directly from recorded runs; the supplement includes scripts to reproduce summary tables, agreement, and the invalid taxonomy.

Appendix: audit artifact inventory (supplement)

Versioned inputs. Prompt variants (02_prompt_variants/), evaluation protocol and judge-contract (03_evaluation_rules/), and task set (01_experiment_design/).

Run artifacts. Each run directory stores: (1) config (model/decoding/prompt-set versions), (2) raw outputs, (3) judged records (per-dimension scores + evidence), and (4) summary tables. Invalid evaluations are stored under 04_results/02_cross_model_evaluation/invalid_evaluations/ and are included in the taxonomy analysis.

Reproduce tables/figures. From supplement/supplement_min/:

```
python 03_evaluation_rules/compute_scores.py
python tools/invalid_to_taxonomy.py
```

Anonymization note. For double-blind review, keep the public code link out of the main PDF; include either (a) an anonymized repository link or (b) a zipped supplement with code + artifacts.

References

[1] OpenReview. Formatting Instructions for ICLR 2026 Conference Submissions.
[2] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, Graham Neubig. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, 2023.
[3] Jiawei Gu, Xuhui Jiang, Zhichao Shi, et al. A Survey on LLM-as-a-Judge. arXiv:2411.15594, 2024.