PROMPT-LEVEL DRIFT AS AN OPERATIONAL MONITORING PROBLEM: SCHEMA FAILURE CLIFFS AND JUDGE-VERSION RISK IN ARTIFACT-GROUNDED EVALUATION

Yuchen Zhu
Shanghai University
zyc20070222@gmail.com

ABSTRACT

Prompt templates are routinely edited in deployed LLM systems and evaluation pipelines, often treated as "non-functional" refactors. We reframe *prompt-level drift* as an *operational monitoring* problem: minimal prompt perturbations can trigger discontinuous schema violations, degrade instruction following, and even shift conclusions when the *judge prompt*—the measurement instrument—changes. Methodologically, we study these failures through an artifact-grounded evaluation pipeline built on a fixed evidence snapshot, where preserved outputs, versioned judge/model slices, processed records, and summary tables form a one-way audit chain. Across six preserved judge/model slices under a baseline judge (three judge models scoring outputs from the other two providers), reducing contract salience from explicit to implicit increases schema hard-breaks (A_structure$=0$) from $8/48$ (16.7%) to $34/48$ (70.8%) and drops the mean total score from 7.69 to 2.75 (max total = 10). We further show judge-version sensitivity on *identical preserved outputs*: for one overlapping slice (ChatGPT-as-judge scoring Gemini outputs under implicit triggers), the hard-break rate shifts from $8/8$ (v0) to $0/8$ (v1/v2), and the mean total score shifts from 0.0 (v0) to 8.0 (v1) to 10.0 (v2). These patterns motivate CAO-style change control: pin prompt and judge versions, monitor interface validity, and prohibit pooling metrics across judge versions.

1 INTRODUCTION

LLM applications commonly rely on prompts not only to specify the task, but also to enforce output *interfaces* required by downstream tooling (e.g., fixed sections, JSON-like blocks, or other schemas). In practice, prompts are edited frequently: teams reorder constraints, "clean up" formatting, or paraphrase evaluation rubrics. A tacit assumption is that these edits are safe—that behavior and measured quality change smoothly.

This paper challenges that assumption from an engineering perspective. We study *prompt-level drift* as an operational risk surface with three incident types: (i) **interface incidents** (schema failures that break parsers), (ii) **quality incidents** (instruction-following degradation under minimal edits), and (iii) **measurement incidents** (score shifts caused by judge prompt changes rather than generator changes). Our goal is *not* to propose a new evaluator or a more robust prompt. Instead, we aim to make prompt drift *monitorable and auditable* in a Catch–Adapt–Operate setting.

Methodologically, we ground this framing in an artifact-grounded evaluation pipeline (Figure 1): prompt variants produce preserved raw outputs, judges produce versioned judge/model slices, and reporting is derived through processed records and summary tables. This one-way evidence chain makes prompt-drift incidents auditable without re-running generators or judges and turns qualitative prompt edits into monitorable operational events.

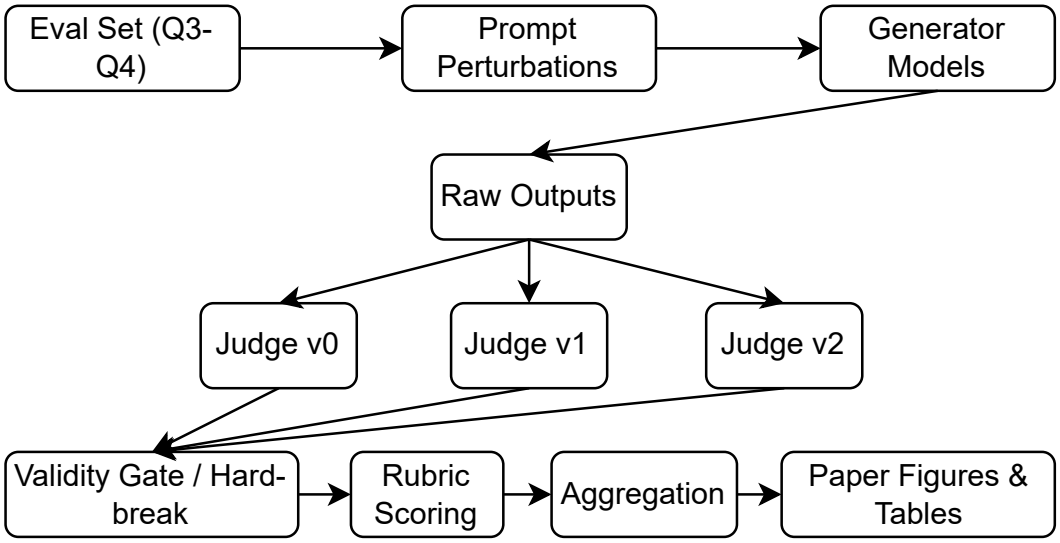

Figure 1: **Evaluation pipeline overview.** A visual summary of the one-way evidence chain from prompt variants to raw outputs, versioned judge/model slices, processed records, and summary tables.

Contributions.

- **Operational framing of prompt drift.** We treat prompt edits as configuration drift and define incident-level outcomes (schema hard-breaks, dimension-level degradation, judge-version sensitivity).
- **Artifact-grounded evaluation pipeline.** We introduce a one-way evidence chain from prompt variants to preserved outputs, versioned judge/model slices, processed records, and summary tables, enabling auditability without re-running generators or judges.
- **Operational risk patterns under minimal perturbations.** Using preserved outputs and versioned judges, we surface schema failure cliffs, broad rubric-dimension degradation, and judge-version sensitivity as concrete monitoring risks.

2 Related Work

We connect three practice-facing threads: LLM-as-judge reliability, prompt robustness/interface contracts, and drift monitoring.

LLM-as-judge reliability. LLM evaluators can be biased and prompt-sensitive, motivating controls and debiasing (Zheng et al., 2023; Chiang et al., 2024; Dubois et al., 2024; Chen et al., 2024; Mukherjee, 2025; Feuer et al., 2025). We treat judge prompting as *instrumentation drift* and emphasize an evidence chain from prompts to outputs to reported summaries.

Prompt robustness and interface contracts. Robustness work studies sensitivity to surface perturbations (Zhu et al., 2023); we focus on non-adversarial refactors that break downstream interfaces, operationalized via schema hard-breaks (A_structure$=0$).

Monitoring under drift. Monitoring emphasizes slice-aware diagnostics and change control (Kang et al., 2020; Yao et al., 2022; Baby et al., 2025; Hao et al., 2025); CAO suggests prompts and judges be versioned dependencies with guardrails. We contribute concrete, auditable failure patterns under minimal edits.

3 PROBLEM SETUP: PROMPT DRIFT AS OPERATIONAL RISK

3.1 PROMPT EDITS AS CONFIGURATION DRIFT

We define **prompt-level drift** as any change to generator or judge prompts that preserves the intended task but changes surface form-wording, ordering, markers, delimiters, or emphasis. Operationally, such edits are common during maintenance (policy updates, refactors, localization, "make it cleaner") and can be shipped without model changes. Prompt drift is therefore a first-class monitoring input: it can create incidents even when the model and data are stable.

3.2 OUTPUT CONTRACT AND SCHEMA HARD-BREAKS

Our evaluated generator prompts require a fixed **three-section output contract**. The rubric includes a structure dimension A_structure scored in $\{0, 1, 2\}$. We define a **schema hard-break** as A_structure$=0$, meaning the output does not satisfy the interface contract (e.g., missing required sections or invalid ordering). Hard-breaks are operationally meaningful because they represent *interface outages*: downstream systems cannot safely consume the output regardless of content quality.

3.3 INCIDENT TAXONOMY AND MONITORING TARGETS

We monitor prompt drift through three incident targets, each derived from the reported metrics:

- **Interface validity:** schema hard-break rate $\mathrm{HBR}(\pi)$ (Eq. 3) and section-level validity gates.
- **Quality degradation:** dimension means $\mu_d(\pi)$ (Eq. 4), dimension failure rates $\mathrm{DFR}_d(\pi)$ (Eq. 5), and mean total $\mathrm{MT}(\pi)$ (Eq. 2).
- **Measurement drift:** judge-version deltas on preserved outputs $\Delta_m^{\text{judge}}(\cdot)$ (Eq. 6).

Deterministic monitoring metrics over a fixed evaluation snapshot. Let $\mathcal{R}$ be the set of preserved evaluation records for a fixed run. A *slice* π is any conjunction of filters over record metadata (e.g., `trigger_type`, `prompt_variant`, `generator_model`, `question_id`, `judge_version`). We use the following quantities as *bookkeeping summaries* over the fixed snapshot, without distributional assumptions. Let $\mathcal{D} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}\}$ denote the five rubric dimensions: A (structure), B (snapshot constraint), C (actionability), D (completeness), E (drift failure). Let v denote the judge prompt version (`judge_version`).

$$n(\pi) := |\mathcal{R}_\pi| \tag{1}$$

$$\mathrm{MT}(\pi) := \frac{1}{n(\pi)} \sum_{i \in \mathcal{R}_\pi} \mathrm{total}_i \tag{2}$$

$$\mathrm{HBR}(\pi) := \frac{1}{n(\pi)} \sum_{i \in \mathcal{R}_\pi} \mathbf{1}[A\_structure_i = 0] \tag{3}$$

$$\mu_d(\pi) := \frac{1}{n(\pi)} \sum_{i \in \mathcal{R}_\pi} d_i, \quad d \in \mathcal{D} \tag{4}$$

$$\mathrm{DFR}_d(\pi) := \frac{1}{n(\pi)} \sum_{i \in \mathcal{R}_\pi} \mathbf{1}[d_i < d^{\max}], \quad d \in \mathcal{D}, \text{ where } d^{\max} = \max(\mathcal{S}_d) = 2 \tag{5}$$

$$\Delta_m^{\text{judge}}(j \parallel j_0; \pi) := M_m(\pi, v{=}j) - M_m(\pi, v{=}j_0) \tag{6}$$

In Eq. 6, m can be `total` or any dimension score; $M_m(\pi, v{=}j)$ denotes the same slice mean operator as in Eq. 2 restricted to records with judge version $v = j$ (with `total` replaced by m).

These correspond to CAO operational questions: *Catch* incidents early, *Adapt* prompts with guardrails, and *Operate* with stable instrumentation.

4 Evaluation Protocol and Evidence Base

4.1 Evidence-as-authority principle

This paper treats preserved evaluation records and fixed scoring rules as evidence; all evidence needed for review appears in this PDF. We do not re-run generators or judges, and we do not introduce new experiments or new measurements. All quantitative claims in this paper are computed from the fixed snapshot and summarized in Figures 1–4 and Table 1.

- per-record evidence: row-level evaluation records
- grouped aggregates (deterministic summaries used for reporting and plotting)

Note. The processed summaries used for reporting are derived artifacts generated from preserved evaluation records under fixed scoring rules.

Development vs. evaluation slices. Earlier slices (Q1–Q2) were used only for prompt/judge iteration and are excluded from all analyses. All quantitative claims are based solely on frozen evaluation slices (Q3–Q4).

4.2 One-way evidence chain

We maintain an explicit one-way evidence chain that supports auditability:

$$\text{prompt variants} \rightarrow \text{raw outputs} \rightarrow \text{judge slices} \rightarrow \text{processed records} \rightarrow \text{summary tables}.$$

Later stages are derived views and must not mutate earlier artifacts. This design makes quantitative claims auditable: reported aggregates can be expanded to row-level records, which in turn link back to preserved raw outputs and versioned judge/model slices.

4.3 Normative vs. derived artifacts

The protocol distinguishes:

- **Normative rules:** scoring dimensions, validity gates, and prompt variant definitions.
- **Derived views:** long-form and grouped CSV summaries used for reporting and plotting.

Operationally, this separation prevents "metric drift by editing the rules after the fact". A CAO workflow should treat normative rules as change-controlled configuration, and treat derived views as rebuildable reports.

5 Results

Numeric source. Unless stated otherwise, results are computed from processed summaries under the baseline judge version v0. We report both counts (for hard-breaks) and means (for total and dimension scores), and we do not pool across judge versions. The underlying evidence corresponds to a single-pass factorial sweep over preserved conditions (provider $\times$ question $\times$ variant $\times$ trigger), not repeated i.i.d. samples; denominators such as "48" indicate preserved evaluation records per trigger type (here: 6 judge/model slices $\times$ 8 preserved outputs), not repeated sampling. Accordingly, we avoid sampling-controlled significance claims and emphasize descriptive effect sizes, worst-slice behavior, and directional consistency across matched conditions.

5.1 Schema failure cliffs under minimal perturbations

Reducing contract salience from explicit to implicit creates a discontinuity in interface validity. Across six preserved judge/model slices under v0 (48 scored files per trigger type; 6 slices $\times$ 8 preserved outputs), the schema hard-break rate jumps from $8/48$ (16.7%) under explicit triggers to $34/48$ (70.8%) under implicit triggers. This is not a gradual degradation; it is a *cliff* in interface validity.

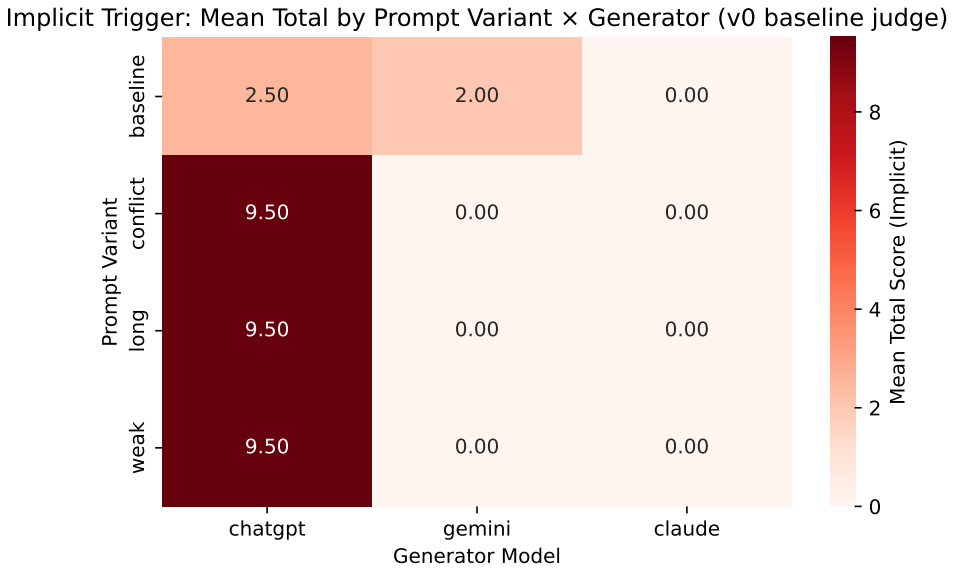

Figure 2: **Implicit-trigger mean total by prompt variant and generator family (v0).** Heatmap shows mean total scores under implicit triggers, highlighting strong heterogeneity across prompt variants and generator families. Numbers computed from the fixed snapshot under v0. Computed as $\mathrm{MT}(\pi)$ (Eq. 2) with `trigger_type=implicit`, grouped by `prompt_variant` and `generator_model`.

This cliff is highly generator-dependent (same v0 evidence, aggregating across judges for each generator family):

- ChatGPT outputs: $0/16$ (explicit) $\rightarrow$ $3/16$ (implicit).
- Claude outputs: $8/16$ (explicit) $\rightarrow$ $16/16$ (implicit).
- Gemini outputs: $0/16$ (explicit) $\rightarrow$ $15/16$ (implicit).

Operationally, prompt drift can therefore create localized outages that are invisible in a single aggregate score. Counts use uniform denominators per generator family (fixed snapshot).

5.2 Instruction-following degradation beyond schema break

Schema failures co-occur with broad quality degradation across rubric dimensions. Under v0, the mean total score drops from 7.6875 (explicit) to 2.7500 (implicit) (max total = 10; fixed snapshot, v0). Dimension means (each in $\{0, 1, 2\}$) also collapse. Let (A, B, C, D, E) denote the rubric dimensions: (A) structure; (B) snapshot constraint; (C) actionability; (D) completeness; (E) drift failure. Dimension-level performance (on a 0–2 scale) reflects this collapse. Under explicit triggers, means across all five dimensions—including structure, actionability, and completeness—ranged from 1.25 to 1.67; under implicit triggers, these dropped to between 0.44 and 0.58. Failure rates (fraction of examples with score < 2) rise sharply across all dimensions. For example, the failure rate for dimension E_drift_failure increases from $14/48$ (29.2%) under explicit triggers to $40/48$ (83.3%) under implicit triggers (fixed snapshot, v0). Figure 3 summarizes dimension-level breakdown and failure rates.

5.3 Slice-level diagnostics: why averages hide outages

Operational monitoring must be slice-aware because failures concentrate. We highlight three concrete slice patterns that are visible only when stratifying by question and prompt variant:

ChatGPT: Slice-Specific Resilience. Under implicit triggers, ChatGPT slices mostly retain high scores: among the eight slices (questions $q3$ and $q4$ across baseline, conflict, long and weak variants), only the $q4$–*baseline* slice fully collapses with mean total $= 0.0$ and a 100% hard-break rate $(2/2)$.

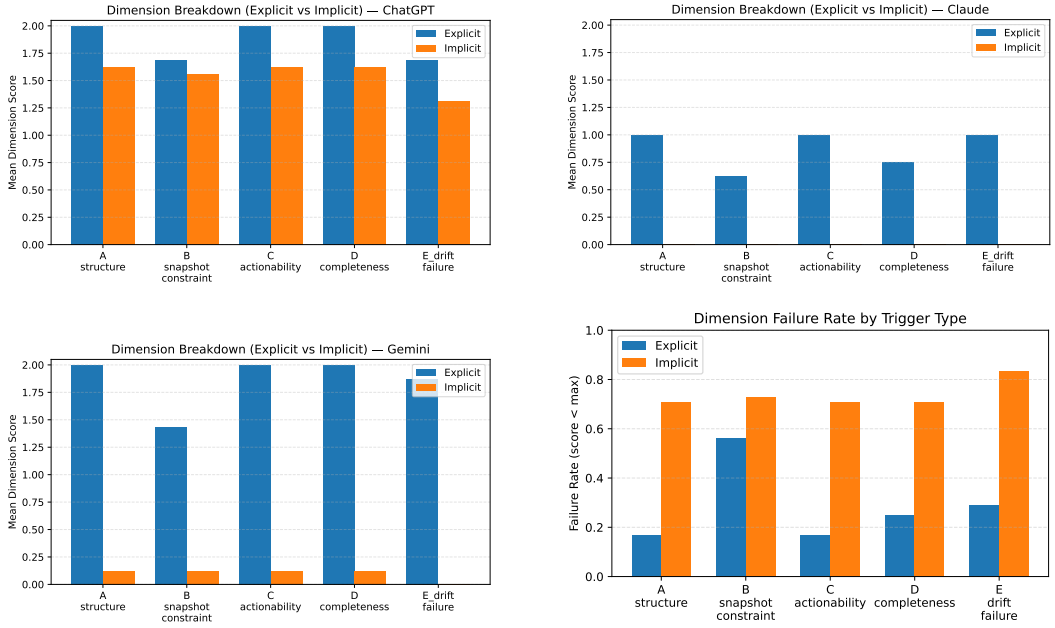

Figure 3: **Dimension-level degradation under implicit triggers (v0).** Mean scores by dimension for ChatGPT, Claude, and Gemini, plus failure rates (score < 2) aggregated across generator families. Numbers are sourced from the fixed snapshot under v0. Top three panels are computed as $\mu_d(\pi)$ (Eq. 4); the failure-rate panel is computed as $\mathrm{DFR}_d(\pi)$ (Eq. 5) over frozen artifacts.

The $q3$–*baseline* slice shows partial degradation (mean total $= 5.0$; hard-break rate $1/2$), while the remaining six slices maintain high total scores (≥ 9) with zero hard-breaks. Thus, contract-salience reduction is catastrophic only on specific slices rather than universally for ChatGPT.

Claude: Systemic Interface Failure. Claude outputs collapse across all slices under implicit triggers: for every combination of question ($q3, q4$) and prompt variant (baseline, conflict, long, weak), the mean total score is 0.0 and the hard-break rate is 100% ($2/2$). Implicit prompts therefore eliminate usable structure for Claude outputs.

Gemini: Context-Dependent Instability. Gemini outputs largely collapse under implicit triggers. Across all eight slices, the mean total score drops to 0.0 and the hard-break rate is 100% ($2/2$) except for the $q4$–*baseline* slice, which retains partial structure (mean total $= 4.0$; hard-break rate $1/2$). This heterogeneity again underscores that contract salience interacts strongly with both the generator and the task/variant context. Operationally, dashboards should report both global aggregates and worst-k slices by hard-break rate and total-score drop.

5.4 Judge-version sensitivity as a measurement incident

Monitoring systems rely on LLM-based judges as instrumentation; judge prompt edits therefore constitute instrumentation migrations that can shift reported severity on identical preserved outputs, and should be interpreted as measurement risk rather than evidence of correctness.

Interpretation rule (instrument migration). We treat each judge prompt version as defining a distinct metric. Therefore: (i) do not compare absolute scores across judge versions without an overlap-based calibration step, and (ii) do not pool or average metrics computed under different judge prompt versions. Operationally, judge upgrades should be gated using overlapping preserved slices to bound expected metric shifts before changing dashboards or alerts.

The fixed snapshot includes multiple judge versions applied to overlapping preserved outputs. We highlight one overlapping slice with complete totals: ChatGPT-as-judge scoring Gemini-generated outputs under implicit triggers.

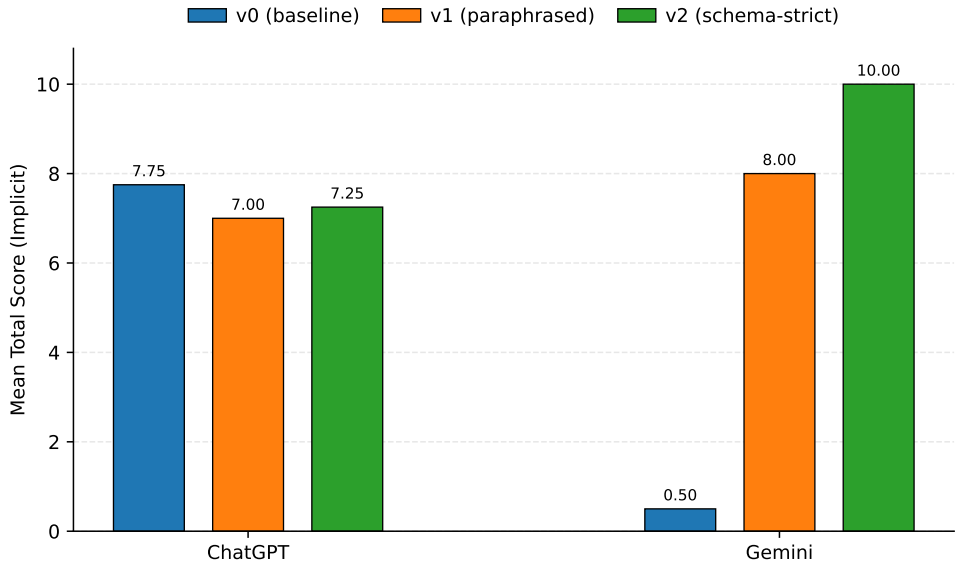

Figure 4: **Judge-version sensitivity on preserved outputs.** Comparison of v0, v1, and v2 under implicit triggers. Numbers sourced from the fixed snapshot. Mean totals are computed as $\mathrm{MT}(\pi)$ (Eq. 2); judge-version effects are summarized by Δ_m^{judge} (Eq. 6).

For this fixed set of 8 preserved outputs:

- v0 baseline judge: hard-breaks $8/8$; mean total = 0.0
- v1 paraphrased judge: hard-breaks $0/8$; mean total = 8.0 (range 7–9)
- v2 schema-strict judge: hard-breaks $0/8$; mean total = 10.0

All values are computed from preserved evaluation records for each judge version; no generator inference is re-run. Figure 4 further summarizes judge-version sensitivity. In the fixed snapshot, v1/v2 are available for two overlapping judge/model slices—ChatGPT evaluating Gemini outputs and Gemini evaluating ChatGPT outputs—while the remaining slices are available only under v0.

6 Operational Implications

We map the observed incident patterns to a CAO workflow. **Catch** corresponds to interface validity and worst-slice monitors (detect outages early); **Adapt** treats prompt edits as migrations with canaries and rollback; **Operate** governs the judge as versioned instrumentation (pinning and no pooling across versions).

6.1 Catch: monitor interface validity and slice-level metrics

The first-line operational signal is **schema validity**. Hard-break rates can be tracked continuously and alerted on as interface incidents. Given strong heterogeneity (Section 5.3), monitoring should also be **slice-aware**: at minimum, stratify by generator family, question, and prompt variant. A practical "incident panel" can be computed directly from row-level evaluation records without new experiments.

6.2 Adapt: treat prompt edits as migrations, not refactors

The schema cliff indicates that "making prompts more natural" by removing explicit markers can cause outages. Two practical adaptation patterns follow: (i) keep contract markers stable and implement stylistic polishing *after* generation (rendering layer), and (ii) deploy prompt edits via canaries with explicit rollback criteria based on schema and dimension-level monitors. When a drift

Signal	Definition (computed over the fixed snapshot)
Hard-break rate	$\mathrm{HBR}(\pi)$ (Eq. 3) by slice (question/variant/model)
Dim. failure rate	$\mathrm{DFR}_d(\pi)$ (Eq. 5) for each dimension d
Mean total	$\mathrm{MT}(\pi)$ (Eq. 2) by slice and overall
Coverage	$n(\pi)$ (Eq. 1) preserved records in slice
Judge version	`judge_version` (pinned evaluator prompt version)

Table 1: Minimal CAO monitoring schema computable from the fixed snapshot (no new experiments; deterministic aggregation only). All signals are computed from preserved evaluation records and their deterministic grouped summaries.

edit is required (policy updates), treat it as a migration with a pinned prompt version, rollback, and a post-mortem if hard-breaks exceed baseline.

6.3 Operate: version and govern the judge as instrumentation

Judge-version sensitivity is a measurement incident: instrument changes can flip conclusions. We recommend three governance rules aligned with CAO:

- **Version pinning**: treat judge prompts as production dependencies with explicit version bumps.
- **No pooling across versions**: each judge version defines a distinct metric; compare only after overlap-based calibration.
- **Migration check**: gate upgrades using overlapping preserved slices to bound expected deltas.

6.4 A minimal monitoring schema

Table 1 lists a minimal monitoring schema computable from the fixed snapshot for CAO-style operations.

7 Limitations

Scope boundary. Our evidence base is intentionally narrow and fixed: we analyze preserved artifacts rather than re-running generators/judges. All quantitative claims are confined to the frozen evaluation slices (Q3–Q4).

Generalization and inference limits. We stop short of claiming statistical significance or i.i.d. generalization beyond these specific slices; counts such as $8/48$ denote matched evaluation records, not repeated samples. We do not claim that any judge version is a ground-truth reference; observed cross-version shifts should be treated as calibration deltas requiring overlap checks.

8 Conclusion

Prompt-level drift is an operational monitoring problem. Using a fixed-snapshot evidence base, we show that small prompt perturbations can trigger schema failure cliffs, degrade rubric-dimension scores, and shift evaluation conclusions when judge prompts change, with substantial slice-level heterogeneity. These findings motivate CAO-style controls: monitor interface validity, treat prompt edits as migrations with rollback criteria, and govern judges as versioned instrumentation.

Reproducibility

Code, prompts, processed evaluation snapshots, and figure-generation scripts are available in our public GitHub repository: `https://github.com/yuchenzhu-research/iclr2026-cao-prompt-drift-lab`. All claims are based on frozen Q3–Q4 slices and the audit chain in Section 4 and Figure 1, and were verified against preserved outputs from the fixed snapshot.

REFERENCES

D. Baby, B. Han, S. Zhang, C. Hu, B. Wang, and Y.-X. Wang. Adapting to online distribution shifts in deep learning: A black-box approach. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

G. H. Chen, S. Chen, Z. Liu, F. Jiang, and B. Wang. Humans or LLMs as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*, 2024.

W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. I. Jordan, J. E. Gonzalez, and I. Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning (ICML)*, 2024.

Y. Dubois, B. Galambosi, P. Liang, and T. B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

B. Feuer, C.-Y. Tseng, A. S. Lathe, O. Elachqar, and J. P. Dickerson. When judgment becomes noise: How design failures in LLM judge benchmarks silently undermine validity. *arXiv preprint arXiv:2509.20293*, 2025.

Wei Hao, Zixi Wang, Lauren Hong, Lingxiao Li, Nader Karayanni, AnMei Dasbach-Prisk, Chengzhi Mao, Junfeng Yang, and Asaf Cidon. Nazar: Monitoring and adapting ml models on mobile devices. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '25), Volume 1*, 2025. doi: 10.1145/3669940.3707246.

D. Kang, D. Raghavan, P. Bailis, and M. Zaharia. Model assertions for monitoring and improving ML models. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

S. Mukherjee. Meta-evaluation collapse: Who judges the judges of judges? OpenReview, 2025. URL https://openreview.net/forum?id=IF0L7HSs3K.

H. Yao, C. Choi, B. Cao, Y. Lee, P. W. Koh, and C. Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. *arXiv preprint arXiv:2211.14238*, 2022.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023.

K. Zhu, J. Wang, J. Zhou, Z. Wang, H. Chen, Y. Wang, L. Yang, W. Ye, Y. Zhang, N. Z. Gong, and X. Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *arXiv preprint arXiv:2306.04528*, 2023.