WORKFLOWPERTURB: Calibrated Stress Tests for Evaluating Multi-Agent Workflow Metrics

Madhav Kanda[1], Sharad Agarwal[2], Rodrigo Fonseca[2], Alok Gautam Kumbhare[2], Pedro Las-Casas[2]

[1]University of Illinois Urbana-Champaign
[2]Microsoft

Abstract

Multi-agent LLM systems that generate structured workflows from natural-language requests are now deployed in production across cloud automation, DevOps, and enterprise process orchestration. Operating such systems exposes a recurring change-management problem. Routine updates, such as re-running the same input, swapping the underlying LLM, or refactoring an agent's prompt or orchestration code, frequently produce workflows that differ substantially from previously validated references. Engineers are then left without a principled way to decide whether a change is safe to ship. Automatic workflow evaluation is the natural tool for answering this question. In practice, however, metric scores are poorly calibrated, and a numeric change rarely communicates the severity of the underlying degradation. We introduce WORKFLOWPERTURB, a controlled benchmark for studying workflow evaluation metrics by applying realistic, graded perturbations to golden workflows. WORKFLOWPERTURB contains 4,973 golden workflows and 44,757 perturbed variants across three perturbation types (Missing Steps, Compressed Steps, and Description Changes), each applied at severity levels of 10%, 30%, and 50%. We benchmark multiple metric families and analyze their sensitivity and calibration using expected score trajectories and residuals. Our results characterize systematic differences across metric families and support severity-aware interpretation of workflow evaluation scores in change-management settings. Our dataset will be released upon acceptance.

1 Introduction

Large Language Models (LLMs) are increasingly used to generate multi-step, dependency-rich workflows for high-stakes settings, where failures can have real operational consequences. Such workflows arise in domains including scientific automation, biomedical information access, question answering, enterprise copilots, and cloud and DevOps systems (Bran et al., 2024; Jin et al., 2024; Microsoft, 2026b; Chandrashekar, 2025; Dalal et al., 2026). In these settings, correctness depends not only on surface-level textual similarity, but on the presence of required steps and the correctness of their ordering and dependencies.

Such workflows are produced by multi-agent systems deployed in production (Microsoft, 2026b; Chandrashekar, 2025). Sustained operation of one such deployed multi-agent workflow generation system, LLexus (Las-Casas et al., 2024), over more than two years surfaces an engineering problem largely orthogonal to model accuracy. Routine system changes frequently produce workflows that differ substantially from validated references. Three classes of change are particularly disruptive: (i) re-running the same request under stochastic decoding yields a structurally different workflow; (ii) swapping in a newer LLM (sometimes a forced upgrade when older models are deprecated (Microsoft, 2026a)) shifts the distribution of generated steps; and (iii) refactoring an agent's prompts, tool inventories, or orchestration code alters which steps are emitted and how they are sequenced. Each forces engineers to decide, before shipping, whether the new workflow is functionally equivalent to its reference or is a silent regression that could cause an outage. Manual review does not scale. Automatic workflow metrics are the natural alternative, but they proved poorly calibrated in practice, with numeric changes rarely communicating the *severity* of degradation. This experience motivated the controlled, generic benchmark presented here, decoupled from any specific production system.

Evaluating these workflows remains unexpectedly difficult. Existing workflow metrics each capture only a narrow slice of correctness: lexical metrics fail under paraphrasing, structural metrics may overlook semantic drift, and semantic metrics

can assign favorable scores to workflows missing critical steps. A central practical challenge is calibration: it is often unclear what a score means in terms of functional risk. For example, does a drop from 0.90 to 0.84 reflect harmless rephrasing or the loss of an essential dependency? Without calibration, metric values are hard to use for regression testing, model comparison, or automated filtering.

The consequences of getting this wrong are severe. Flawed orchestration can cause outages or misconfigured infrastructure, and subtle workflow deviations can alter analytical conclusions in scientific and biomedical domains. This sharpens the practical question: how can we determine whether a modified workflow remains functionally equivalent to a validated reference, and integrate such checks into CI/CD without manual review?

We introduce WORKFLOWPERTURB, a benchmark of over 44,000 workflows constructed through controlled perturbations. We generate three perturbation types, Missing Steps, Compressed Steps, and Description Changes, each applied at graded intensity levels of 10%, 30%, and 50%. These structured degradations enable systematic analysis of metric calibration under known severity, revealing which metrics are robust to benign edits, which fail under functional degradation, and how to interpret workflow scores reliably.

2 Related Work

Agent and workflow evaluation. Recent benchmarks evaluate LLMs in agentic and tool-using settings, where models must select, invoke, and sequence external functions: e.g., the Berkeley Function Calling Leaderboard (Patil et al., 2025) emphasizes end-to-end task success in tool-augmented environments. A complementary line of work analyzes execution trajectories to diagnose and localize agent failures, including AGENTRX (Barke et al., 2026) (intermediate actions, tool calls, and environment feedback) and TRAIL (Deshpande et al., 2025) (trace-based reasoning over structured action sequences). Both families operate on runtime traces and assume access to environment interactions or success signals; in contrast, we study static workflow representations and evaluate how automatic metrics respond when workflows deviate from golden references in controlled and graded ways, enabling systematic analysis of metric calibration, sensitivity, and blind spots. Adaptive execution frameworks such as VineLM (Pagonas et al., 2026) select a different LLM at each configurable workflow stage under per-request cost and latency budgets. They further expand the space of outputs a single workflow can produce across runs, sharpening the need for calibrated cross-run metric comparisons of the kind we study here.

LLM-as-a-Judge and Its Limitations. LLMs have emerged as scalable automatic evaluators with strong agreement to human judgments (Zheng et al., 2023; Chiang and Lee, 2023), but recent work highlights important reliability failures. Shi et al. (2025) systematically study position bias across 15 judges and 22 tasks and find that judgments depend strongly on candidate ordering and on the quality gap between candidates. These are properties of how options are *presented* to the judge, rather than of the underlying artifact. Related work also documents biases such as length, self-preference, and stylistic effects (Wang et al., 2024). These studies characterize fragility along one axis: *perturbing the inputs shown to the judge*. Our perspective is complementary and orthogonal: we hold the judge's prompt and presentation fixed and instead perturb the *artifact under judgment* at calibrated severities, measuring how a judgment-based metric responds to known functional degradation.

3 WORKFLOWPERTURB

We model a workflow as a directed acyclic graph (DAG) $G = (V, E)$, where each node $v \in V$ represents a step described in natural language and each edge $(u, v) \in E$ denotes a precedence constraint. Given a golden workflow G and a candidate workflow G', an evaluation metric produces a score $s(G, G') \in [0, 1]$ (or a normalized variant) intended to reflect workflow quality or similarity. Rather than proposing a new workflow generator, our focus is to analyze *metric behavior*: under controlled degradations of known severity, do metric scores change in an interpretable and calibrated manner? We study this question by introducing structured perturbation types, graded severity levels, and an expected score trajectory characterizing ideal metric response under degradation.

3.1 Benchmark Construction

Source Data. We begin with the set of workflows in WORFBENCH (Qiao et al., 2025) and retain those with $|V| \geq 5$ nodes (**at least** five steps). This lower threshold ensures each workflow has enough steps to meaningfully perturb and evaluate metric

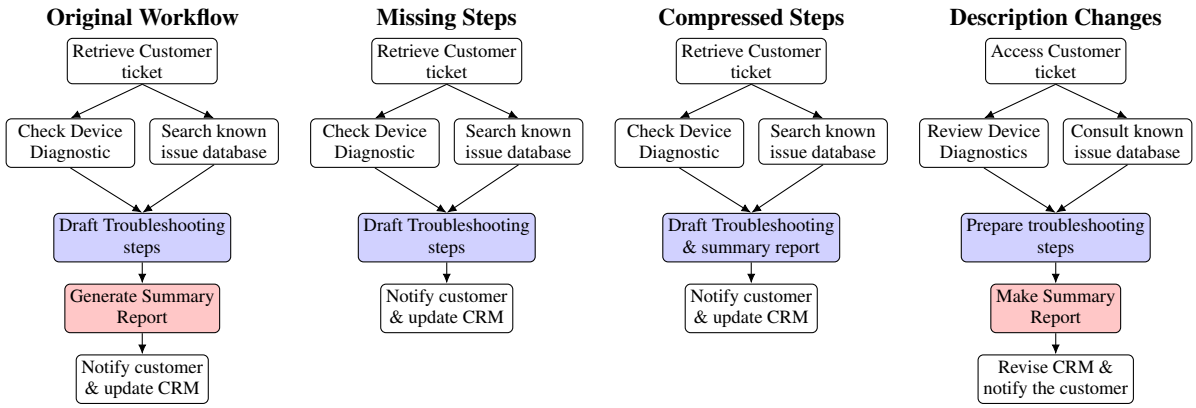

Figure 1: Real-world example illustrating perturbations in LLM-generated workflows for a customer support escalation task. MISSING STEPS: The red node present in the original workflow is omitted. COMPRESSED STEPS: The red and blue nodes are merged into a single blue node, reducing granularity. DESCRIPTION CHANGES: All nodes retain the same semantic meaning as the original but differ in syntactic phrasing.

sensitivity. We impose no upper cap, so the corpus spans short-to-long workflows: 5–6–41 nodes (min–median–max), with 13 workflows containing 20 or more nodes. The resulting corpus contains 4,973 golden workflows drawn from seven source datasets (WikiHow, ALFWorld, WebShop, Lumos, ToolBench, OS, and ToolAlpaca; WikiHow and ALFWorld together account for 86% of the corpus). Combined with three perturbation types at three severity levels, this yields 44,757 perturbed variants in the benchmark we will release.

Perturbation Patterns. From our 2+ years operating LLM-generated workflows in production, three recurring degradation patterns emerged. We model them as controlled perturbations applied to golden workflows, illustrated on a customer-support escalation task in Figure 1:

- **Missing Steps:** LLM-generated workflows often omit essential actions that are required for complete task execution. Such omissions result in incomplete or truncated execution paths when compared to the golden workflow.
- **Compressed Steps:** Generated workflows frequently contain fewer nodes than their golden counterparts, condensing multiple fine-grained actions into a single broader step. While this may preserve most of the content, it alters the structural properties of the workflow and leads to mismatches under graph-based evaluation.
- **Description Changes:** Even when workflows preserve the correct structure, step descriptions often differ syntactically from the golden reference. This lexical variation confuses surface metrics while preserving semantics.

Diagnostic mapping. These three patterns are not redundant along the metric axis (the seven metrics named below are formally defined in Section 4). MISSING STEPS most distinctly stresses structural (Graph/Chain F1) and judgment-based metrics by reducing required nodes; COMPRESSED STEPS stresses both structural and ordering metrics (Kendall's τ) by collapsing edges and reshuffling precedence; and DESCRIPTION CHANGES isolates lexical metrics (BLEU, GLEU) because graph topology is preserved by construction. This makes each perturbation type a controlled diagnostic for a different metric family (see Section 5).

To study the impact of these issues on evaluation metrics, we apply each perturbation type at three severity levels $\{10\%, 30\%, 50\%\}$ of the golden workflow (an illustration for MISSING STEPS appears as Figure 4 in the appendix). These graded levels mimic realistic variations and let us probe metric calibration, i.e., whether scores degrade proportionally with severity.

Perturbation Generation and Validation. For each workflow and perturbation configuration, we employ LLMs (GPT-4o as the primary generator, with GPT-4.1 and o3-mini as fallbacks for variants that initially fail the static checks below) to generate variants. To ensure correctness, each variant undergoes automated static validation checks:

- **Node Count Consistency:** Given a golden workflow with 10 nodes, applying a 10% removal

perturbation must produce a workflow with precisely 9 nodes. Any deviation from this expected count is flagged as inconsistent.

- **Change Count Verification:** For a 30% perturbation on a 10-node workflow, exactly 3 nodes must be altered (removed, compressed, or paraphrased, depending on type). If fewer or more are modified, the output is rejected. For removal and compression, verification compares node counts before and after. For paraphrasing under the primary path, verification counts how many step descriptions changed; the LLM chooses which N nodes to paraphrase under that constraint.

These checks target the two most common generation failure modes: (i) deleting or merging too many nodes, and (ii) over- or under-applying paraphrasing. Each generator call prepends a fixed few-shot example (one per perturbation type) before the row-specific instruction: the golden workflow plus the perturbation directive. Variants that fail validation enter a feedback-driven refinement loop, escalating to stronger models on repeated failure. The loop is effective: the initial GPT-4o pass left 17.7%, 27.5%, and 44.7% of Missing-Steps, Compressed-Steps, and Description-Changes variants failing the static checks, and successive fallbacks reduced these to 0.03%, 0.14%, and 1.3% (4, 21, and 200 residual variants, respectively). Prompt templates, the per-split handling of these residuals, and the description-change generation pipeline appear in Appendix A. We also manually inspected a random subset of variants to confirm perturbations were applied as intended (e.g., removed nodes were genuinely removed and paraphrases preserved factual content), rather than to measure human–LLM agreement. A small number of residual variants were hand-corrected; see Appendix A.7 for details.

Score Assignment and Perturbation Alignment. Each workflow is assigned a score reflecting perturbation severity, with lower scores indicating deviation from the golden workflow. For *Missing Steps* and *Compressed Steps*, the score equals the remaining workflow fraction, computed as $1 - p$ for perturbation percentage p (e.g., p=30% $\rightarrow$ 0.7, p=50% $\rightarrow$ 0.5). In the case of Compressed Steps, this decrease is justified by the loss of structural granularity: multiple fine-grained actions (often corresponding to distinct tool invocations) are merged into broader nodes, violating the one-tool-per-step abstraction and reducing fidelity to the golden execution graph. For *Description Changes*, the score remains constant across severity levels, since these edits affect only textual descriptions and not the underlying workflow structure.

4 Metrics

We evaluate seven metrics across five families. Structural metrics (Graph F1, Chain F1) measure topological consistency via subgraph matching and sequence alignment. Lexical metrics (BLEU, GLEU) compute n-gram overlap. BERTScore measures contextual embedding similarity. Kendall's τ evaluates rank-order consistency. LLM-as-Judge provides rubric-based holistic assessment. Full definitions appear in Appendix B.

4.1 Structural Metrics

Chain F1. Following Qiao et al. (2025), predicted node chains are aligned with valid topological orderings of the golden workflow, and the longest increasing subsequence (LIS) is computed. Precision and recall are defined as $p_{\text{chain}} = l/|V_{\text{pred}}|$ and $r_{\text{chain}} = l/|V_{\text{golden}}|$, where l is the LIS length. The final score is their harmonic mean (F1).

Graph F1. The predicted and golden workflow graphs are aligned using a Maximum Common Induced Subgraph (MCIS) algorithm. If the largest shared subgraph contains k nodes, precision and recall are defined as $p_{\text{graph}} = k/|V_{\text{pred}}|$ and $r_{\text{graph}} = k/|V_{\text{golden}}|$. The score is their harmonic mean (F1).

4.2 Lexical Metrics

BLEU (Papineni et al., 2002) is a precision-oriented n-gram matching metric with a brevity penalty, while **GLEU** (Wu et al., 2016) balances n-gram precision and recall and is more robust for shorter or paraphrased step descriptions.

4.3 Semantic Metric

BERTScore (Zhang et al., 2020) compares contextual embeddings of step descriptions from pre-trained transformer models and computes precision, recall, and F1 via token-level cosine similarity.

4.4 Ordering Metric

Kendall's τ (Kendall, 1938) measures rank correlation between generated and gold step orderings, defined as $\tau = (C - D)/\left(\frac{1}{2}n(n-1)\right) \in [-1, 1]$, where C and D are the numbers of concordant and discordant pairs among n aligned steps.

4.5 LLM-as-Judge

To complement automatic metrics, we use an *LLM-as-Judge* evaluation with GPT-4o (Azure OpenAI). The model is prompted with the task description, golden workflow, generated workflow, and a rubric defining 0–5 scores covering correctness, completeness, ordering, and clarity, with anchor examples.

5 Results and Analysis

We evaluate metric behavior across perturbation types and severities, assigning each workflow a predefined degradation score for interpretability. All reported *LLM-as-Judge* scores are linearly normalized from the original 0–5 scale to the [0,1] range for comparability across metrics.

Overall trends. Across all perturbation types, metric scores generally decrease as perturbation severity increases. This confirms that the benchmark introduces progressively harder deviations from the golden workflows. However, degradation patterns differ substantially across metric families, as detailed below. Appendix Table 5 condenses these into a 7×3 sensitivity matrix Δ_m^{avg}, defined in Section 5.1.

Missing Steps. Removing steps (dropped red node in Figure 1; numbers in Table 2) disrupts completeness and dependency structure. Structural metrics degrade nearly linearly with the removal rate, closely tracking imposed severity, while lexical metrics drop more sharply under token loss. BERTScore is the most tolerant, with a mild and slightly non-monotonic decline between 30% and 50%. Kendall's τ and LLM-as-Judge both decline strongly, the latter signaling missing content.

Compressed Steps. Merging fine-grained actions into broader steps (cf. the merged red/blue nodes in Figure 1; numbers in Table 3) substantially degrades structural scores and Kendall's τ, the latter reflecting disrupted pairwise precedence under compression. Lexical metrics decline more than under deletion, since merged descriptions break direct n-gram overlap. BERTScore is already low at mild compression and falls only slightly, suggesting embedding similarity is itself sensitive to abstraction. LLM-as-Judge tracks the loss of procedural granularity despite preserved intent.

Description Changes. When only descriptions are modified (paraphrased-but-isomorphic graph in Figure 1; numbers in Table 4), graph structure remains intact and structural metrics, Kendall's τ, and LLM-as-Judge stay nearly flat. Lexical metrics decline with heavier paraphrasing, as expected under surface-level variation. BERTScore stays high, unlike under structural perturbations, indicating that semantic similarity is largely preserved.

Overall, no single metric captures all failure modes: structural and ordering metrics detect missing or compressed steps but miss textual edits, while lexical metrics respond strongly to paraphrasing yet over-penalize deletions; embedding and judgment metrics are complementary.

5.1 Sensitivity Analysis

To quantify how rapidly each metric degrades with severity, we summarize the per-severity score curves by a single *sensitivity* Δ_m^{avg}:

$$\Delta_m^{\text{avg}} = \frac{1}{2}\left[\frac{\bar{s}_m(10\%)-\bar{s}_m(30\%)}{0.20} + \frac{\bar{s}_m(30\%)-\bar{s}_m(50\%)}{0.20}\right] \quad (1)$$

where $\bar{s}_m(\alpha)$ is the mean score of metric m at $\alpha\%$ perturbation (appendix Tables 2–4). Figure 3 visualizes Δ_m^{avg}; Appendix Table 5 gives the numbers.

Structural metrics (Graph/Chain F1). Peak under *compressed steps* ($\Delta = 1.04$); moderate for *missing steps* (0.72); low for *description changes* (0.29). Both metrics primarily track topological distortion (cf. Figure 1).

Lexical metrics (BLEU, GLEU). High across all perturbation types ($\Delta \approx 0.84$–1.23), peaking for *missing steps* where token loss dominates. They cannot, however, distinguish structural loss from benign rewording on their own.

Embedding metric (BERTScore). Least sensitive overall ($\Delta < 0.4$ everywhere): tolerant of structural change (0.24 for *missing steps* and 0.23 for *compressed steps*) and only modestly responsive to *description changes* (0.39).

Order-based metric (Kendall's τ). Strongest for *compressed steps* (1.42); high for *missing steps* (0.94); essentially flat for *description changes* (0.02), consistent with unchanged topology.

Judgment-based metric (LLM-as-Judge). Tracks structural/order behavior: high for *missing steps* (0.80) and moderate for *compressed steps* (0.70); near zero, and slightly negative within sample noise, for *description changes* (-0.03).

Per-severity score tables appear in Appendix C.

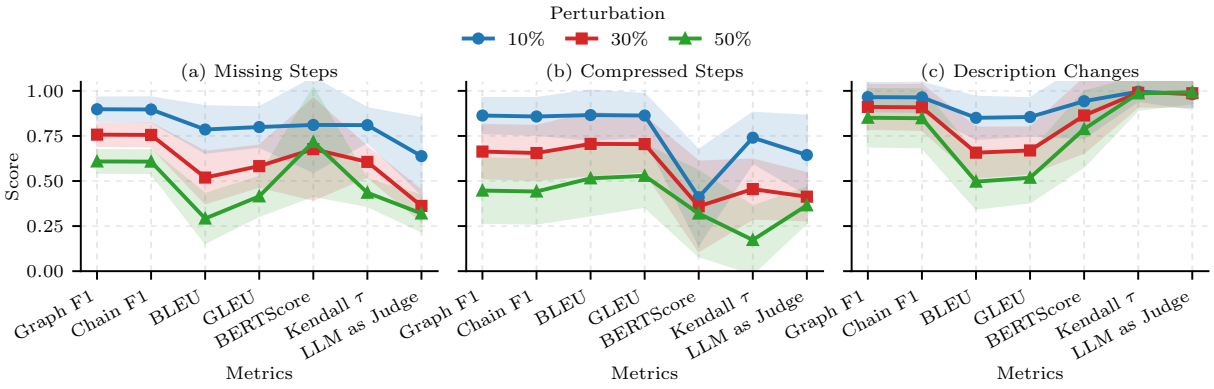

Figure 2: Metric scores by perturbation type and severity (mean ± std).

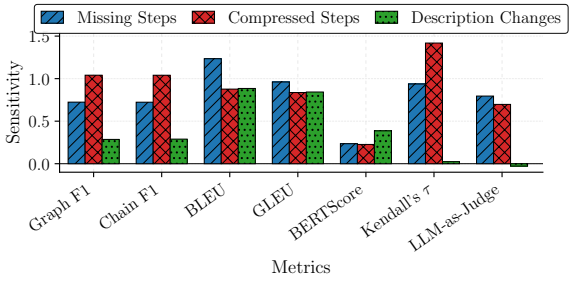

Figure 3: Per-metric sensitivity Δ_m^{avg} across the three perturbation types.

Table 1: Recommended metric bundle and alert thresholds per failure mode.

Failure mode	Primary	Secondary	Alert
Missing Steps	Graph/Chain F1	LLM-Judge	< 0.75
Compressed Steps	Kendall's τ	Graph/Chain F1	< 0.65
Description Changes	BLEU/GLEU	BERTScore	< 0.70

6 Key Insights and Practical Guidance

Workflow validation under change. Deployed workflows evolve with model upgrades, prompt revisions, and orchestration changes, and metric families respond differently to structural versus textual edits (Section 5). WORKFLOWPERTURB enables severity-aware calibration: post-change candidates are compared against approved goldens using a calibrated bundle (Table 1), whose assignments and thresholds derive from the sensitivity matrix and per-severity distributions in Appendix C. Since no single metric suffices, breaches can trigger review or rollback, prioritizing structural regressions.

Cost, latency, and open-weight judges. LLM-as-Judge dominates pipeline cost: at public GPT-4o pricing ($2.50/M input, $10/M output tokens) each call runs ∼$0.005–0.01 and seconds of latency. At CI/CD scale this is non-trivial. The cheaper bundle metrics can therefore pre-filter, reserving the judge for ambiguous cases. WORKFLOWPERTURB also supports benchmarking open-weight judges and learned workflow comparators.

7 Conclusion

We introduced WORKFLOWPERTURB, a benchmark for evaluating workflow metrics under controlled structural and textual perturbations. It was motivated by recurring change-management failures in a multi-agent workflow system we have run in production for over two years. Our analysis shows that metrics have distinct sensitivity patterns and calibration gaps across failure modes, so no single uncalibrated score is a sound basis for CI/CD shipping decisions.

We expect WORKFLOWPERTURB to help other teams that operate multi-agent workflow generators and face the same change-management question: whether a re-run, model swap, or prompt or orchestration change has silently regressed a validated workflow. To use it, a team expresses its golden and candidate workflows in the same node/edge form, scores candidates against approved references with the same metric families, and applies the calibrated bundle and thresholds (Table 1) to gate CI/CD changes. The thresholds are operating points, not fixed constants: they can be re-tuned to a team's tolerance for missed regressions versus false alerts, and the perturbation protocol can be re-run on in-domain goldens to recalibrate.

Limitations

Scope of evaluation. WORKFLOWPERTURB evaluates workflows as static graph and text artifacts: we measure how metric scores respond to controlled perturbations but do not correlate calibration with downstream execution success, since no shared, reproducible sandbox can faithfully execute workflows across the heterogeneous domains and tool stacks in the corpus. We treat metric calibration as a necessary prerequisite for execution-aware evaluation rather than a substitute for it. We also study only three perturbation families (MISSING STEPS, COMPRESSED STEPS, DESCRIPTION CHANGES) at three severities; addition, duplication, and reordering of steps are not covered, so our calibration claims may not extrapolate to those failure modes.

Data and judge coverage. The golden workflows are inherited from WORFBENCH (Qiao et al., 2025) (English-only, filtered to $|V| \geq 5$) and are concentrated at the short end (median 6 nodes; only 13 of 4,973 exceed 20). Generalization to long industrial workflows is therefore untested, and the 50% setting acts on very small graphs for most of the corpus. Our LLM-as-Judge results use a single judge (GPT-4o via Azure OpenAI at temperature 0); we do not benchmark open-weight or alternative proprietary judges, and reproducibility depends on its continued availability and version stability.

Pipeline and metric choices. Perturbed variants are produced by LLMs and accepted by an automated validation pipeline with only spot-check human review, so residual artifacts may persist in unreviewed portions of the 44,757-variant corpus. Structural metrics (Graph/Chain F1) use a fixed alignment threshold ($\tau_{\text{align}} = 0.8$) and encoder that were not swept; absolute scores would shift under other settings, though the relative ordering used for sensitivity (§5.1) is the robust signal of interest. Finally, we scope the study to seven widely deployed off-the-shelf metrics and do not compare against learned or task-specific comparators, leaving that to future work on the same open benchmark.

Ethical Considerations

Data provenance and licensing. The golden workflows underlying WORKFLOWPERTURB are derived from WORFBENCH (Qiao et al., 2025), a publicly released benchmark; we inherit its license terms and use its English-language workflow set unchanged modulo the $|V| \geq 5$ filter. All 44,757 perturbed variants are produced synthetically by LLMs (GPT-4o as the primary generator, with GPT-4.1 and o3-mini as fallbacks for variants that initially failed automated validation) following the controlled protocol described in §3. A small number of variants were hand-corrected by the authors: 4 rows in the MISSING STEPS split, for which all automated regeneration attempts continued to fail the static checks. These rows are flagged as manually authored in the dataset we will release. No new human-subjects data was collected, and no personally identifying information is present in the corpus.

Intended use and dual-use risks. WORKFLOWPERTURB is intended as an offline diagnostic resource for calibrating evaluation metrics used to validate LLM-generated workflows, for example inside CI/CD regression-test pipelines for deployed multi-agent systems. The benchmark itself does not generate, execute, or recommend production actions; it scores already-produced workflow artifacts against a reference. We therefore see low direct dual-use risk. A practitioner who relied *solely* on an uncalibrated metric score for a production ship/no-ship decision, however, could ship a regression silently; the central finding of the paper (Section 5) is precisely that this is unsafe, which is why we recommend the calibrated bundle in Table 1 rather than any single score.

Compute, cost, and reproducibility. LLM-as-Judge evaluations use GPT-4o via Azure OpenAI at temperature 0. Reproducing the judge-based numbers therefore depends on continued availability of GPT-4o and is subject to provider retirement policies; this is also noted in the Limitations section. All other metrics (Graph/Chain F1, BLEU, GLEU, BERTScore, Kendall's τ) use open implementations and are deterministic given the same input. To make the cost trade-off transparent, we report per-evaluation cost estimates from public OpenAI pricing in §6.

AI writing assistance. We used generative AI tools only for polishing of authored text. All technical content, experimental design, analysis, and conclusions are the authors' own; no AI tool was used to generate experimental results or citations.

References

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. 2026. Agentrx: Diagnosing ai agent failures from execution trajectories. *Preprint*, arXiv:2602.02475.

Andres M Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535.

Kirankumar Chandrashekar. 2025. Streamline operational troubleshooting and tasks with amazon q developer cli. AWS DevOps Blog. Posted 2025-06-19.

Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Dwip Dalal, Madhav Kanda, Zhenhailong Wang, Heng Ji, and Unnat Jain. 2026. Compositional reasoning via joint image and language decomposition. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5753–5775, Rabat, Morocco. Association for Computational Linguistics.

Darshan Deshpande, Varun Gangal, Hersh Mehta, Jitin Krishnan, Anand Kannappan, and Rebecca Qian. 2025. Trail: Trace reasoning and agentic issue localization. *Preprint*, arXiv:2505.08638.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2024. Genegpt: augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics*, 40(2):btae075.

Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Harold W. Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97.

Pedro Las-Casas, Alok Gautam Kumbhare, Rodrigo Fonseca, and Sharad Agarwal. 2024. LLexus: An AI agent system for incident management. *ACM SIGOPS Operating Systems Review*, 58(1):23–36.

Microsoft. 2026a. Azure openai in azure ai foundry models model retirements. Microsoft Learn documentation.

Microsoft. 2026b. Create a cloud flow in power automate using copilot. Microsoft Learn documentation. Last updated 2026-01-16.

Nikos Pagonas, Matthew Lou, Tianyi Peng, Dan Rubenstein, and Kostis Kaffes. 2026. VineLM: Trie-based fine-grained control for agentic workflows. *Computing Research Repository*. ArXiv:2605.23914.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2025. Benchmarking agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Lin Shi, Chiyu Ma, Wenhua Liang, Xingjian Diao, Weicheng Ma, and Soroush Vosoughi. 2025. Judging the judges: A systematic study of position bias in LLM-as-a-judge. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 292–314, Mumbai, India. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, and 1 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Preprint*, arXiv:1609.08144.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang,

Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Track on Datasets and Benchmarks*.

Appendix

In this appendix, we include the following to supplement the main paper:

A. Prompt Templates and Generation Pipeline (Section A). Exact LLM prompts used to generate perturbed workflow variants, including error-injection and refinement strategies, the static-check failure trajectory and residual-failure handling, and the description-change generation pipeline.

B. Detailed Metric Definitions (Section B). Formal definitions, equations, and implementation details for all evaluation metrics.

C. Extended Results and Sensitivity Analysis (Section C). Complete metric tables across perturbation types and a detailed analysis of metric sensitivity under controlled degradation.

A Prompt Templates and Generation Pipeline

The following prompts were used to generate perturbed workflow variants. Placeholders N denote the exact number of nodes to alter, computed before the prompt is assembled as $N = \max(1, \lceil |V| \times \alpha \rceil)$ for severity $\alpha \in \{0.1, 0.3, 0.5\}$ (see §3.1). The prompts below show the instruction template; each call also embeds the concrete golden workflow.

A.1 Few-Shot Examples (prepended to every generator call)

For each perturbation type, every generator call prepends a fixed few-shot example before the row-specific instruction. The row-specific portion consists of the original golden workflow plus the perturbation directive, in the form shown in Section A.2 below.

Missing Steps.

```
Original workflow:
Node:
1: go to potential locations where a
kettle may be found
2: take kettle from where it was found
. . .
7: put kettle in/on storage.
Edge: (START,1) (1,2) . . . (6,7) (7,END)
Please remove one step from the workflow
above and return the new workflow in the
same Node:/Edge: format.
Perturbed (one step missing):
Node:
1: go to potential locations where a
kettle may be found
```

```
. . .
6: open storage if necessary
Edge: (START,1) (1,2) . . . (5,6) (6,END)
—
```

Compressed Steps.

```
Original workflow:
Node:
1: locate the mug in the kitchen
2: pick up the mug
3: go to the coffee machine
4: place the mug under the coffee spout
5: press the button to brew coffee
Edge: (START,1) (1,2) (2,3) (3,4) (4,5)
(5,END)
Please compress two steps in the workflow
above (combine them into fewer steps)
and return the new workflow in the same
Node:/Edge: format.
Perturbed (two steps compressed):
Node:
1: locate the mug in the kitchen and
pick it up
2: go to the coffee machine
3: place the mug under the coffee spout
and press the button to brew coffee
Edge: (START,1) (1,2) (2,3) (3,END)
—
```

Description Changes.

```
Original workflow:
Node:
1: go to the living room
2: find the remote control
3: turn on the television
4: select the news channel
Edge: (START,1) (1,2) (2,3) (3,4)
(4,END)
Please paraphrase exactly two steps
in the workflow above (change their
description but not their meaning). Do
not change more or fewer than two steps.
Keep all other node descriptions exactly
the same. Return the new workflow in the
same Node:/Edge: format.
Perturbed (two steps paraphrased):
Node:
1: go to the living room
2: locate the remote control device
3: turn on the television
4: switch to the news channel
Edge: (START,1) (1,2) (2,3) (3,4)
(4,END)
—
```

A.2 Row-Specific Prompt Templates

In the templates below, {golden workflow} denotes the concrete golden workflow inserted into the prompt, and N is the exact number of nodes to alter, defined above.

A.3 Missing Steps Prompt

```
Original workflow: {golden
workflow}
Please remove exactly N steps
from the workflow above and
return the new workflow in the
same Node:/Edge: format.
Perturbed (N steps missing):
```

A.4 Compressed Step Prompt

```
Original workflow: {golden
workflow}
Please compress exactly N steps
in the workflow above (combine
them into fewer steps) and return
the new workflow in the same
Node:/Edge: format.
Perturbed (N steps compressed):
```

A.5 Description Changes Prompt

```
Original workflow: {golden
workflow}
Please paraphrase exactly N
steps in the workflow above
(change their description but not
their meaning). Do not change
more or fewer than N steps.
Keep all other node descriptions
exactly the same. Return the new
workflow in the same Node:/Edge:
format.
Perturbed (N steps paraphrased):
```

A.6 Iterative Refinement and Escalation

When a generated output fails validation, the variant enters a feedback-driven refinement loop in which the validation error is re-injected into the next prompt. For example:

```
Your output had 8 nodes instead
of 9 for a 10% removal. Please
regenerate the workflow and
ensure that exactly one node is
removed. Return only the modified
workflow as a numbered list of
steps.
```

If the corrected output again fails validation (e.g., paraphrases 4 nodes instead of 3), the process is repeated up to three times per model. After three unsuccessful iterations, the task is escalated to a stronger generation model.

A.7 Residual-Failure Handling

The body reports the static-check failure trajectory of the iterative refinement loop. The raw

initial-pass counts are 2,634/14,919 for Missing Steps, 4,106/14,919 for Compressed Steps, and 6,673/14,919 for Description Changes, and the successive fallbacks were GPT-4.1, an Azure-OpenAI content filter, and finally o3-mini with feedback-driven re-prompting. The few residual variants that survived were then resolved per split. The 4 residual MISSING STEPS variants that continued to fail automated regeneration were hand-corrected by the authors and are flagged as manually authored in the released dataset. The 21 residual COMPRESSED STEPS variants were blocked by an Azure OpenAI content filter on the perturbed workflow text and were regenerated through a standard OpenAI API endpoint. The 200 residual DESCRIPTION CHANGES variants were repaired by the hybrid fallback in Appendix A.8.

A.8 Description-Change Generation Pipeline

Description-change variants follow the same primary path as the other perturbation types: a bulk workflow-editing prompt (few-shot plus row-specific instruction) asks the LLM to paraphrase exactly N step descriptions while preserving graph structure, with up to three static-check retries and subsequent escalation to stronger models with feedback-driven re-prompting. For the ≈ 200 variants that still failed after escalation, we applied a *hybrid* fallback: N node indices are selected at random, a generator LLM paraphrases each selected step description in a separate call, and the workflow is reassembled with edges unchanged. This fallback is more reliable than single-shot whole-workflow editing.

B Detailed Metric Definitions

We present the formal definitions and implementation details of all evaluation metrics used in WORKFLOWPERTURB. Where applicable, we include equations and illustrative examples showing how the scores are computed.

B.1 Chain F1

Idealized objective. Suppose the predicted node chain is $C(V^p)$ and the gold workflow graph is $G(V^g, E^g)$. From G, we can enumerate all possible topological orderings $\{C(V^g)_1, C(V^g)_2, \ldots\}$. The predicted chain is compared against each ordering using the *Longest Increasing Subsequence (LIS)*:

$$l_i = \mathrm{LIS}(C(V^g)_i, C(V^p)),$$

and the longest valid subsequence length is taken as

$$l = \max(|l_1|, |l_2|, \ldots, |l_n|).$$

Precision and recall are then

$$p_{\text{chain}} = \frac{l}{|V^p|}, \qquad r_{\text{chain}} = \frac{l}{|V^g|},$$

yielding

$$f_1^{\text{chain}} = \frac{2\, p_{\text{chain}}\, r_{\text{chain}}}{p_{\text{chain}} + r_{\text{chain}}}.$$

Implementation. Enumerating every topological ordering of G is exponential in $|V^g|$ in the worst case, and is further complicated by the fact that node identities are not shared across workflows. The predicted and gold node sets differ in surface form and must first be aligned semantically. We therefore use the following polynomial-time approximation:

1. Embed each step description (gold and predicted) with Sentence-BERT (Reimers and Gurevych, 2019), using the `all-mpnet-base-v2` encoder.
2. Compute the cosine-similarity matrix between predicted and gold step embeddings, and solve an optimal one-to-one assignment with the Hungarian algorithm (Kuhn, 1955) (`scipy.optimize.linear_sum_assignment`).
3. Retain only assigned pairs whose cosine similarity is $\geq \tau_{\text{align}} = 0.8$; unmatched predicted steps are dropped.
4. Let σ be the sequence of gold indices that the retained predicted steps map to, taken in predicted-chain order. Compute $l = |\mathrm{LIS}(\sigma)|$ under the natural gold ordering.

This realizes the LIS objective against a *single* canonical gold ordering (the order in which the gold graph's steps are listed), rather than against all topological orderings. The substitution is exact for the linear-chain workflows that dominate our corpus (where the topological ordering is unique) and is an approximation for the small number of multi-branch workflows.

Threshold and encoder. Both the encoder choice and the alignment threshold τ_{align} are fixed across all experiments to keep relative comparisons across perturbations meaningful. The threshold of 0.8 was chosen because in the `all-mpnet-base-v2` embedding space sentence pairs with cosine similarity below this value are

typically only loosely related, while values above it correspond to paraphrastic or near-paraphrastic agreement. We did not run a full sweep over τ_{align}; this dependence is acknowledged in the Limitations section.

B.2 Graph F1

Idealized objective. Given predicted graph $G(V^p, E^p)$ and gold graph $G(V^g, E^g)$, we seek the Maximum Common Induced Subgraph (MCIS):

$$G_{mcis}(V_{mcis}, E_{mcis}) = \text{MCIS}\Big(G(V^p, E^p), G(V^g, E^g)\Big).$$

If the largest matched subgraph contains $k = |V_{mcis}|$ nodes, then

$$p_{\text{graph}} = \frac{k}{|V^p|}, \qquad r_{\text{graph}} = \frac{k}{|V^g|},$$

and the final score is

$$f_1^{\text{graph}} = \frac{2\, p_{\text{graph}}\, r_{\text{graph}}}{p_{\text{graph}} + r_{\text{graph}}}.$$

This procedure captures both node correctness and edge consistency, providing a stricter evaluation than simple edge overlap.

Implementation. Exact MCIS on labeled graphs is NP-hard. As in Chain F1, the predicted and gold node sets do not share identifiers and must be aligned semantically before any subgraph notion makes sense. We approximate the MCIS objective as follows:

1. Compute the same SBERT (`all-mpnet-base-v2`) embeddings, cosine-similarity matrix, and Hungarian one-to-one assignment as in Chain F1, and retain only pairs with cosine similarity $\geq \tau_{\text{align}} = 0.8$. Call the retained alignment π and let $S = \pi(V^p) \subseteq V^g$.
2. Map every predicted edge $(u, v) \in E^p$ through π, producing a set of induced edges on S.
3. Iteratively prune S: remove any node $v \in S$ whose neighbor set induced from the mapped predicted edges within S does not equal its neighbor set in the gold graph restricted to S. Repeat until no further removal is possible.
4. Let $k = |S|$ after pruning and compute p_{graph}, r_{graph}, f_1^{graph} as above.

The pruning step enforces the *induced-subgraph* condition: a node is retained only if its neighborhood within S is identical under both the gold and the mapped predicted edge sets. The output is therefore a common induced subgraph of G^p and G^g under the alignment π. It is not necessarily the maximum one, since π is fixed up front by the embedding-based assignment rather than searched jointly with S. The same encoder and threshold as in Chain F1 apply, with the same caveat about τ_{align} sensitivity.

B.3 BLEU and GLEU

BLEU (Papineni et al., 2002) measures n-gram precision between candidate and gold step descriptions, with a brevity penalty to discourage very short outputs. GLEU (Wu et al., 2016) balances precision and recall over n-grams, making it more robust to short sequences. In our implementation, all step descriptions of a workflow are concatenated (whitespace-joined, in order) into a single document, and the score is computed once per workflow against the corresponding gold concatenated document using NLTK's `sentence_bleu` (with `SmoothingFunction().method1`) and `sentence_gleu`, with `nltk.word_tokenize` tokenization. Workflow-level scores are then averaged across the dataset to produce the means reported in Table 5 and Appendix C.

B.4 BERTScore

BERTScore (Zhang et al., 2020) measures the semantic similarity of step descriptions using contextual token embeddings. Our implementation calls the public `bert_score` library[1] with `lang="en"` and `rescale_with_baseline=True`; under these settings the library selects `roberta-large` as the encoder and rescales each token-level similarity by the corpus-baseline similarity between unrelated sentence pairs. We report the rescaled F1, averaged across the aligned step pairs within a workflow and then across the dataset. The baseline rescaling makes the absolute scores noticeably lower than the un-rescaled BERTScore values typically reported in the MT/summarization literature; the quantity of interest in this paper is the *relative* change in BERTScore across perturbation severities, which is unaffected by the shift.

Alignment of unequal step lists. Since perturbations such as *Missing Steps* and *Compressed Steps*

[1] https://github.com/Tiiiger/bert_score

change the number of steps, the gold and perturbed step lists are not the same length. BERTScore is fundamentally a paired-list metric (item i on the candidate side is compared with item i on the reference side), so some alignment policy is required before it can be invoked. We use a deliberately simple and conservative rule: order-preserving truncation to $\min(|gold|, |pred|)$, comparing the first $\min$ step descriptions on each side. This choice keeps BERTScore as a *pure semantic-similarity* signal that complements, rather than duplicates, the other metrics in our bundle. Specifically: (i) the cost of *deletions* is already accounted for by the structural metrics (Graph and Chain F1), so introducing a length-mismatch penalty inside BERTScore would double-count missing steps; and (ii) optimal-assignment alignments (e.g., Hungarian matching over embedding similarity) would re-import the very signal that node-cosine similarity and the embedding-based Kendall's τ alignment in the bundle are designed to provide, again collapsing the complementarity between metrics. Since our perturbations operate on contiguous subsets and preserve the order of the remaining steps, the truncated prefix is a faithful slice of the perturbed workflow for semantic comparison.

B.5 Kendall's τ

Kendall's τ (Kendall, 1938) measures the ordinal correlation between two ranked lists. For workflows, we treat the step ordering as a ranking and compute

$$\tau = \frac{C - D}{\frac{1}{2}n(n-1)},$$

where n is the number of aligned steps being compared, and C and D denote the numbers of concordant and discordant step pairs, respectively. τ ranges from -1 (complete disagreement) to 1 (perfect agreement).

B.6 LLM-as-Judge Prompts

For the LLM-as-Judge evaluation, we designed a rubric-based prompt that instructs the model to compare a generated workflow against a gold reference. Workflows are formatted with explicit node and edge lists.

B.6.1 Rubric

- 0 = Completely Incorrect or Irrelevant
- 1 = Major Flaws (breaks core logic)
- 2 = Significant Issues (omits several key steps or adds wrong steps)
- 3 = Noticeable Errors (minor reorder or one extra/missing non-critical step)
- 4 = Very Good (only one minor step missing)
- 5 = Perfect Match (fully adheres, detailed, accurate)

B.6.2 Anchor Examples

We provide concrete anchors to calibrate the model's scoring:

- Example A (Score 4): Workflow missing one minor step.
- Example B (Score 5): Workflow perfectly matches the gold.
- Example C (Score 3): Workflow with minor reordering of steps.
- Example D (Score 2): Workflow with extra irrelevant steps.
- Example E (Score 0): Workflow completely incorrect relative to task.

B.6.3 Prompt Format

The model is instructed to:

1. List nodes and edges from both workflows.
2. Match nodes by content.
3. Identify missing, extra, or reordered steps.
4. Assign an initial score using the rubric.
5. Perform a self-check: confirm if the initial score strictly follows the rubric.
6. Adjust to a final score if necessary.

B.6.4 Expected Output

The model must return exactly one JSON object with the fields:

```
{
  "ThoughtChain": "<step-by-step reasoning>",
  "initial_score": <int 0-5>,
  "self_check": {
    "consistent": "Yes"|"No",
    "final_score": <int 0-5>
  },
  "justification": "<one-sentence summary>"
}
```

Table 2: Average metric scores (mean ± std) at different perturbation levels for Missing Steps.

Metric	Perturbation Level		
	10%	30%	50%
Graph F1	0.90±0.06	0.76±0.06	0.61±0.06
Chain F1	0.90±0.07	0.76±0.06	0.61±0.06
BLEU	0.79±0.13	0.52±0.14	0.29±0.13
GLEU	0.80±0.11	0.58±0.11	0.42±0.11
BERTScore	0.81±0.26	0.68±0.28	0.72±0.30
Kendall's Tau	0.81±0.09	0.61±0.08	0.44±0.07
LLM-as-Judge	0.64±0.21	0.36±0.08	0.32±0.10

Table 3: Average metric scores (mean ± std) at different perturbation levels for Compressed Steps.

Metric	Perturbation Level		
	10%	30%	50%
Graph F1	0.86±0.10	0.66±0.15	0.45±0.18
Chain F1	0.86±0.10	0.66±0.15	0.44±0.18
BLEU	0.87±0.14	0.71±0.17	0.51±0.20
GLEU	0.86±0.12	0.70±0.15	0.53±0.17
BERTScore	0.41±0.26	0.36±0.25	0.32±0.24
Kendall's Tau	0.74±0.14	0.46±0.16	0.17±0.19
LLM-as-Judge	0.64±0.22	0.41±0.13	0.37±0.10

Table 4: Average metric scores (mean ± std) at different perturbation levels for Description Changes.

Metric	Perturbation Level		
	10%	30%	50%
Graph F1	0.97±0.08	0.91±0.12	0.85±0.16
Chain F1	0.96±0.08	0.91±0.12	0.85±0.16
BLEU	0.85±0.12	0.66±0.14	0.50±0.15
GLEU	0.86±0.10	0.67±0.13	0.52±0.14
BERTScore	0.94±0.20	0.86±0.21	0.79±0.21
Kendall's Tau	1.00±0.05	0.99±0.08	0.99±0.09
LLM-as-Judge	0.98±0.08	0.99±0.07	0.99±0.06

Table 5: Metric sensitivity Δ_m^{avg} (per-row mean degradation rate, Section 5.1) across all metrics and perturbation types. Markers: ▲ high (≥ 0.8), ● moderate (0.3–0.8), ○ low (< 0.3).

Metric	Missing	Compressed	Description
Graph F1	0.72 ●	1.04 ▲	0.29 ○
Chain F1	0.72 ●	1.04 ▲	0.29 ○
BLEU	1.23 ▲	0.88 ▲	0.88 ▲
GLEU	0.96 ▲	0.84 ▲	0.84 ▲
BERTScore	0.24 ○	0.23 ○	0.39 ●
Kendall's τ	0.94 ▲	1.42 ▲	0.02 ○
LLM-as-Judge	0.80 ▲	0.70 ●	−0.03 ○

B.6.5 Implementation Notes

We use GPT-4o via Azure OpenAI with temperature 0.0 to minimize randomness. Each workflow pair is evaluated by three independent calls of the same prompt template; the three per-call scores are averaged to absorb residual non-determinism from the inference backend. We do not vary the prompt template across calls.

C Results Analysis

Note on reproducing Table 5 from these tables. The sensitivity values Δ_m^{avg} reported in Table 5 are computed from the *unrounded* per-severity means, whereas the tables above display those means rounded to two decimal places. Applying the formula $\Delta_m^{\text{avg}} = 2.5 \cdot [\bar{s}_m(10\%) - \bar{s}_m(50\%)]$ to the rounded values in Tables 2–4 will therefore reproduce Table 5 only up to ± 0.02 per cell; this difference is rounding noise, not a discrepancy in the underlying data.

Deriving the metric bundle and alert thresholds (Table 1). The Primary/Secondary assignments and operating-point thresholds in the practical-guidance bundle are read directly off the sensitivity matrix (Table 5) and the per-severity score tables (Tables 2–4); they are operating-point recommendations rather than tuned hyperparameters.

Primary/Secondary selection. For each failure mode we take as **Primary** the metric family that is both highly sensitive to that perturbation *and* diagnostically specific (comparatively unresponsive to the unrelated modes), and as **Secondary** a complementary family that covers the Primary's blind spot. (i) MISSING STEPS: structural Graph/Chain F1 ($\Delta = 0.72$) localizes node loss, paired with LLM-as-Judge ($\Delta = 0.80$) for semantic completeness; lexical metrics, although numerically more sensitive ($\Delta_{\text{BLEU}} = 1.23$), are not chosen as Primary because they fire almost equally on all three modes (0.88 on both Compressed and Description) and are therefore non-specific. (ii) COMPRESSED STEPS: Kendall's τ has the largest sensitivity ($\Delta = 1.42$) and is near-inert under Description changes (0.02), making it a specific ordering probe, paired with structural F1 ($\Delta = 1.04$). (iii) DESCRIPTION CHANGES: lexical BLEU/GLEU are the only family that moves while structure and order stay flat (Kendall's τ $\Delta = 0.02$, structural 0.29), paired with BERTScore ($\Delta = 0.39$) to bound whether paraphrasing altered meaning.

Threshold selection. Each alert threshold is an

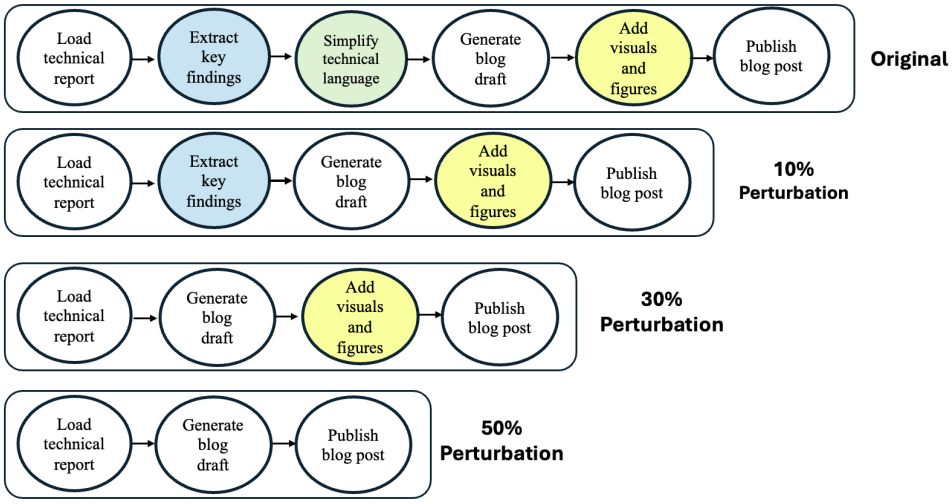

Figure 4: Visualization of MISSING STEPS perturbations applied to the six-node blog generation workflow. At each level, a proportional number of nodes is randomly dropped: 1 node (10%), 2 nodes (30%), and 3 nodes (50%). Higher perturbation levels result in increasingly incomplete workflows that diverge from the golden sequence.

operating point placed in the separating band between the Primary metric's mean score under *mild* (10%) and *moderate* (30%) perturbation. Candidates statistically consistent with $\leq 10\%$ degradation therefore pass, while those reaching $\sim 30\%$ degradation or worse are flagged. For MISSING STEPS, Graph/Chain F1 falls $0.90 \rightarrow 0.76 \rightarrow 0.61$ across $10/30/50\%$, so the trip point is set at 0.75 (the 30% level). For DESCRIPTION CHANGES, BLEU/GLEU fall $\approx 0.85 \rightarrow 0.66 \rightarrow 0.50$, giving a threshold of 0.70, just above the 30% mean. For COMPRESSED STEPS, Kendall's τ drops steeply and with high variance ($0.74 \pm 0.14 \rightarrow 0.46 \pm 0.16$); the threshold is set more conservatively at 0.65, between the 10% and 30% means, to catch fast-onset ordering regressions earlier. These values are defaults intended to be re-tuned to a deployment's own tolerance for missed regressions versus false alerts.