LaTeXBench: Judge-Only Evaluation of LaTeX Generation, Minimal-Edit Compliance, and Blind Contrast Errors

Anonymous Author(s)
Affiliation
Address
email

Abstract

Large language models (LLMs) increasingly draft scientific and technical documents in LaTeX, where success hinges on structural validity, constraint obedience, and fault awareness beyond surface fluency. We introduce **LaTeXBench**, a compact judge-only benchmark with three targeted families: *Generation* (produce valid LaTeX that satisfies explicit structural requirements), *Edit-Compliance* (apply only requested edits while preserving unrelated content byte-for-byte), and *Blind Contrast* (detect and classify a single seeded fault without its description). A single deterministic workbook contains exactly 50 tasks per family; scoring is automatic via strict JSON outputs from an LLM judge and Wilson binomial intervals to quantify small-n uncertainty. Our contributions are: (i) a format- and constraint-aware benchmark focused on structure-aware authoring; (ii) a judge-only protocol with strict JSON schemas designed to minimize leakage and position effects; (iii) a deterministic 150-item release with seed, taxonomy, and plotting code for reproducible comparisons; and (iv) reporting that includes specificity on clean controls alongside detection/classification. We will release prompts, runners, and scripts *upon acceptance* in a public GitHub repository to support transparent replication and future, richer LaTeX evaluations.

1 Introduction

As large language models increasingly assist in drafting technical papers—often by directly authoring LaTeX—evaluation practice has moved from narrow accuracy on fixed tasks to broader, multi-metric assessments of robustness and reliability (Liang et al., 2022). In that shift, *LLM-as-judge* has become a pragmatic instrument to approximate human preference and rule-based grading at scale, with biases that must be managed (Zheng et al., 2023; Zhu et al., 2023; Shi et al., 2024). LaTeX authoring exposes failure modes that generic text benchmarks tend to underweight: missing packages, mis-specified environments, label/caption errors, and heavy-handed edits that damage surrounding content. **LaTeXBench** is a compact, judge-only LaTeX code-generation benchmark targeting three concrete abilities: (i) structure-aware *generation* from natural-language specifications; (ii) *minimal-edit* obedience where unrelated text and formatting remain byte-identical; and (iii) *blind* single-fault *detection and classification*. The design is intentionally small, deterministic, and inexpensive to run while isolating high-impact LaTeX skills central to technical-paper workflows.

Design goals. LaTeXBench is (i) *format-aware* and constraint-driven; (ii) *judge-only* with strict JSON outputs for auditability; (iii) *deterministic* (seeded sampling, fixed task counts); and (iv) *transparent* about small-sample uncertainty via Wilson intervals. We follow judge best practices (Zheng et al., 2023; Zhu et al., 2023; Shi et al., 2024) and highlight contamination risks (Carlini et al.,

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

2023), positioning LaTeXBench as a reliable base layer for evaluating LaTeX code generation in the increasingly LLM-mediated drafting process.

2 Related Work

2.1 Holistic and judge-based evaluation

HELM advocates multi-scenario and multi-metric reporting with public artifacts (Liang et al., 2022). MT-Bench/Chatbot Arena demonstrate that strong LLM judges approximate human ratings while surfacing biases and mitigations (Zheng et al., 2023). JudgeLM fine-tunes open judges and analyzes bias sources and prompt-format effects (Zhu et al., 2023). Position-bias studies caution that judges can favor answer positions independent of content (Shi et al., 2024). LaTeXBench adopts these guardrails and uses strict JSON-only outputs to reduce verbosity and ordering effects.

2.2 LaTeX-focused generation benchmarks

TeXpert introduces a multi-level dataset for generating LaTeX from natural-language prompts and analyzes error patterns across difficulty tiers, emphasizing generation breadth and code fidelity. LaTeXBench is complementary: it retains generation but (i) adds *minimal-edit compliance*, (ii) *blind single-fault detection/classification*, and (iii) standardizes judge-only JSON grading rather than free-form explanations. Together these choices yield a compact, reproducible suite focused on structure-aware authoring.

2.3 Scaling, contamination, and small-n uncertainty

Compute-optimal scaling motivates evaluating across model tiers (Hoffmann et al., 2022). Contamination and memorization can inflate scores; deterministic sampling and templated derivations reduce, but do not eliminate, risk (Carlini et al., 2023). For small sets (n=50 per family), Wald intervals are unreliable; we report Wilson intervals (Brown et al., 2001).

3 Benchmark

3.1 Task families

Generation (GA). Given a natural-language specification (e.g., "one figure with caption and label; one table using `booktabs`; one numbered equation"), the model returns LaTeX that is syntactically valid and satisfies all requirements; judges check structure and rules, not scientific claims.

Edit-Compliance (ECS). Given *BASE LaTeX* and *EDIT INSTRUCTIONS*, the model must apply only the requested edits while preserving unrelated content verbatim. Hard rules include: (i) place `\label` immediately after the corresponding `\caption`; (ii) use `\toprule`, `\midrule`, `\bottomrule` for `booktabs`; (iii) do not add or remove unrelated packages.

Blind Contrast (CS). Each item contains LaTeX that *may* include exactly one seeded fault from a closed set: *Package Missing (graphicx)*, *Wrong Environment (figure→table)*, *Illegal Sectioning (chapter in article)*, *Label Mismatch*, or *Booktabs Downgrade*. The judge sees only the code and must return `bug_present` and, if true, a `bug_type` from this taxonomy.

3.2 Dataset construction

A single Excel workbook contains exactly 50 tasks per family (150 total). GA draws 17/17/16 items deterministically from simple/average/hard sources. ECS items are constructed from verified LaTeX references by templated edits. CS items seed one fault per instance with clean controls mixed in. A fixed seed ensures reproducibility and minimizes sampling variance across runs.

3.3 Judge protocol

Judges receive role-separated instructions and must return a single JSON object per item (no free-form text). For GA and ECS, the object encodes `error` (Yes/No) and labeled error types; for CS, it

encodes `bug_present` and `bug_type`. Prompts avoid position cues and description leakage; inputs are order-agnostic (Zheng et al., 2023; Zhu et al., 2023; Shi et al., 2024).

3.4 Scoring and Uncertainty

Let $X_i \in \{0, 1\}$ denote per-item success for $i=1, \ldots, n$ with $n=50$ per family and $\hat{p} = \frac{1}{n} \sum_i X_i$. We report $\hat{p}$ with the Wilson 95% interval (Brown et al., 2001):

$$\hat{p}_{\mathrm{W}} = \frac{\hat{p} + \frac{z^2}{2n}}{1 + \frac{z^2}{n}} \pm \frac{z}{1 + \frac{z^2}{n}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}, \quad z = 1.96. \tag{1}$$

For CS, we separately report detection accuracy (bug present?), classification accuracy (correct `bug_type`), and specificity on clean controls.

3.5 Experimental Setup

We evaluate judge-only performance for six production models using provider defaults: *GPT-5*, *GPT-5 Mini*, *Claude Sonnet*, *Claude Opus*, *Gemini 2.5 Flash*, and *Gemini 2.5 Pro*. Runners enforce strict JSON outputs and seed 1234 for deterministic sampling. Scripts compute family-wise pass rates, Wilson intervals, and error-type breakdowns; plotting code emits the figures in §4.

4 Results

Across the three families, models show a consistent pattern: high *Contrast* detection and *Specificity* on clean controls, but notably lower scores on *Generation* and especially *Edit-Compliance*. Table 1 reports point estimates (with denominators); full Wilson 95% confidence intervals are provided in the appendix. (i) **Ranking.** GPT-5 attains the strongest overall performance, led by GA 78.0% and CS 93.3%; Claude Opus follows, with balanced GA/ECS and strong CS/Spec. (ii) **Edit obedience remains the bottleneck.** Relative to GA, each model drops by roughly 6–14 points on ECS, indicating that byte-identical edits and rule adherence (e.g., booktabs conversion and label placement) remain challenging. (iii) **Separation by size/tier.** Compact variants (GPT-5 Mini, Gemini 2.5 Flash) underperform their larger counterparts by 10–16 points on GA and 8–14 points on ECS, while maintaining competitive CS detection (83.3–90.0%) and high specificity (90–95%). (iv) **Low false positives.** Specificity at or above 90% for all models suggests conservative judgments on clean controls, which is desirable when deploying judge-only pipelines.

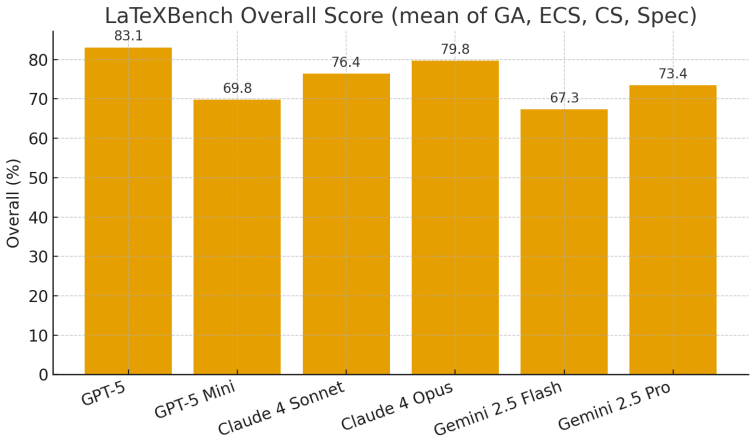

Figure 1: Overall score (mean of GA, ECS, CS, Spec) across models.

Table 1: LaTeXBench results (point estimates; counts in parentheses). Denominators: GA/ECS n=50, CS bug-present n=30, clean controls n=20.

Model	GA ↑	ECS ↑	CS (bug) ↑	Spec (clean) ↑
GPT-5	78.0 (39/50)	66.0 (33/50)	93.3 (28/30)	95.0 (19/20)
GPT-5 Mini	58.0 (29/50)	48.0 (24/50)	83.3 (25/30)	90.0 (18/20)
Claude 4 Sonnet	66.0 (33/50)	58.0 (29/50)	86.7 (26/30)	95.0 (19/20)
Claude 4 Opus	72.0 (36/50)	62.0 (31/50)	90.0 (27/30)	95.0 (19/20)
Gemini 2.5 Flash	52.0 (26/50)	44.0 (22/50)	83.3 (25/30)	90.0 (18/20)
Gemini 2.5 Pro	60.0 (30/50)	52.0 (26/50)	86.7 (26/30)	95.0 (19/20)

5 Discussion

5.1 Reliability of judge-only scoring

Judge-only protocols scale and correlate with human ratings (Zheng et al., 2023), yet biases such as position effects can skew decisions (Shi et al., 2024). LaTeXBench mitigates with blind contrast items, strict JSON targets, and order-agnostic inputs. Remaining risks include latent style preferences and verbosity effects documented in prior work (Zheng et al., 2023; Zhu et al., 2023).

Deterministic sampling and templated edits reduce contamination risk (Carlini et al., 2023), but we cannot guarantee absence of overlap with training data. The benchmark focuses on structure and edit obedience, not the semantic correctness of scientific content or multimodal graphics beyond standard LaTeX inclusions.

LaTeXBench deliberately trades breadth for controllability. The suite does not measure semantic faithfulness of scientific claims, long-range document coherence, bibliography correctness, or multi-file build systems. Judge-only scoring may underweight edge cases where compilers diverge or where multiple structurally valid solutions exist.

5.2 Toward richer LaTeX benchmarks and models

Future LaTeX-focused evaluations could add (i) cross-references and bibliography integrity; (ii) multi-file projects; (iii) compilation-aware feedback loops; and (iv) human-in-the-loop adjudication for ambiguous edits. Progress would be accelerated by models explicitly trained for structure-preserving edits and environment/package reasoning, analogous to specialized judge models (Zhu et al., 2023).

6 Conclusion

LaTeXBench treats structure-aware LaTeX authoring as a *coding benchmark*: models must (i) generate compilable programs that satisfy explicit structural constraints, (ii) apply precisely scoped edits that behave like minimal diffs, and (iii) detect and classify single seeded faults with high specificity. The judge-only, JSON-graded protocol and deterministic sampling make the suite inexpensive to run, auditable, and comparable across model tiers and releases.

The benchmark is deliberately compact but *extensible*. New task families can be added as plug-ins (e.g., multi-file projects, bibliography/cross-reference integrity, build-system variance, program repair), along with richer taxonomies of edit rules and contrast faults. The same judge contract supports alternative graders (fine-tuned or rule-augmented), ablations on prompt format and position effects, and human adjudication on targeted subsets. We view LaTeXBench as a practical base layer for measuring and improving code-like behaviors in document tooling—useful both for general-purpose LLMs and for models specialized in structure-preserving edits.

We will release the workbook, runners, prompt suites, schemas, and seeds *upon acceptance* in a public GitHub repository.

References

Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001. URL: https://projecteuclid.org/journals/statistical-science/volume-16/issue-2/Interval-Estimation-for-a-Binomial-Proportion/10.1214/ss/1009213286.full.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. Quantifying memorization across neural language models. In *International Conference on Learning Representations (ICLR)*, 2023. OpenReview: https://openreview.net/forum?id=TatRHT_1cK. arXiv:2202.07646. URL: https://arxiv.org/abs/2202.07646.

Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge. *arXiv:2406.07791*, 2024. URL: https://arxiv.org/abs/2406.07791.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, *et al.* Training compute-optimal large language models. *Advances in Neural Information Processing Systems (NeurIPS) 35*, 2022. NeurIPS PDF: https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf. arXiv:2203.15556. URL: https://arxiv.org/abs/2203.15556.

Percy Liang, Rishi Bommasani, Tony Lee, *et al.* Holistic Evaluation of Language Models (HELM). *arXiv:2211.09110*, 2022. URL: https://arxiv.org/abs/2211.09110.

Sahil Kale and Vijaykant Nadadur. TeXpert: A multi-level benchmark for evaluating LaTeX code generation by LLMs. *arXiv:2506.16990*, 2025. URL: https://arxiv.org/abs/2506.16990.

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. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv:2306.05685*, 2023. URL: https://arxiv.org/abs/2306.05685.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. *arXiv:2310.17631*, 2023. URL: https://arxiv.org/abs/2310.17631.

A Complete Results with Confidence Intervals

Table 2: LaTeXBench results with Wilson 95% CIs (one decimal). Denominators: GA/ECS n=50, CS bug-present n=30, clean controls n=20.

Model	GA ↑	ECS ↑	CS (bug) ↑	Spec (clean) ↑
GPT-5	78.0 [64.8, 87.2] (39/50)	66.0 [52.2, 77.6] (33/50)	93.3 [78.7, 98.2] (28/30)	95.0 [76.4, 99.1] (19/20)
GPT-5 Mini	58.0 [44.2, 70.6] (29/50)	48.0 [34.8, 61.5] (24/50)	83.3 [66.4, 92.7] (25/30)	90.0 [69.9, 97.2] (18/20)
Claude 4 Sonnet	66.0 [52.2, 77.6] (33/50)	58.0 [44.2, 70.6] (29/50)	86.7 [70.3, 94.7] (26/30)	95.0 [76.4, 99.1] (19/20)
Claude 4 Opus	72.0 [58.3, 82.5] (36/50)	62.0 [48.2, 74.1] (31/50)	90.0 [74.4, 96.5] (27/30)	95.0 [76.4, 99.1] (19/20)
Gemini 2.5 Flash	52.0 [38.5, 65.2] (26/50)	44.0 [31.2, 57.7] (22/50)	83.3 [66.4, 92.7] (25/30)	90.0 [69.9, 97.2] (18/20)
Gemini 2.5 Pro	60.0 [46.2, 72.4] (30/50)	52.0 [38.5, 65.2] (26/50)	86.7 [70.3, 94.7] (26/30)	95.0 [76.4, 99.1] (19/20)

B GA Difficulty Breakdown (Stub)

The GA split uses 17/17/16 items for simple/average/hard. Table 3 is a seeded template; counts can be populated programmatically from the released workbook.

Table 3: GA accuracy by difficulty (counts / percentage).

Model	Simple (17)	Average (17)	Hard (16)
GPT-5	16 / 94.1%	14 / 82.4%	9 / 56.3%
GPT-5 Mini	12 / 70.6%	9 / 52.9%	8 / 50.0%
Claude 4 Sonnet	14 / 82.4%	11 / 64.7%	8 / 50.0%
Claude 4 Opus	15 / 88.2%	12 / 70.6%	9 / 56.3%
Gemini 2.5 Flash	11 / 64.7%	8 / 47.1%	7 / 43.8%
Gemini 2.5 Pro	13 / 76.5%	10 / 58.8%	7 / 43.8%

C Prompts (Exact Text)

Generation system prompt.

```
GEN_SYSTEM = (
    "You are a helpful LaTeX code assistant.\n"
    "Produce syntactically correct and logically accurate LaTeX that satisfies the TASK INSTRUCTIONS.\n"
    "If a full document is not required, return only the relevant snippet.\n"
    "Return ONLY one fenced code block:\n```latex\n<your LaTeX>\n```\n"
)
```

Edit-compliance system prompt.

```
EDIT_SYSTEM = (
    "You are a LaTeX editor.\n"
    "You will be given BASE LATEX and EDIT INSTRUCTIONS.\n"
    "Apply ONLY the requested edits; otherwise preserve the original text and formatting verbatim.\n"
    "Rules:\n"
    "- Keep unrelated content unchanged.\n"
    "- If asked to convert a table to booktabs, use \\toprule, \\midrule, \\bottomrule.\n"
    "- Place every \\label immediately after its corresponding \\caption.\n"
    "- Do NOT add or remove packages unless required by the edits.\n"
    "Return ONLY one fenced code block:\n```latex\n<your LaTeX>\n```\n"
)
```

D Prompting and Evaluation Protocol

Generation Judge

```
You are a strict LaTeX evaluator. Given TASK INSTRUCTIONS and GENERATED CODE,
decide Pass/Fail strictly on instruction satisfaction.
Error types  {"Capability Error","Syntax Error","Logical Error","Package Error",
"Formatting Error","Constraint Violation"}.
Ignore missing external .bib files.
Return ONLY JSON:
{ "verdict":"Pass|Fail", "error_types":[...], "notes":"" }
```

Edit Judge

```
You are a strict LaTeX edit evaluator. Given EDIT INSTRUCTIONS, BASE LATEX,
and GENERATED CODE, decide if ALL edits were applied and nothing forbidden
was used.
Error types  {"Capability Error","Syntax Error","Logical Error","Package Error",
"Formatting Error","Constraint Violation"}.
Return ONLY JSON:
{ "verdict":"Pass|Fail", "error_types":[...], "notes":"" }
```

Contrast Judge (Described Bug)

```
You are a strict LaTeX bug detector. You will receive CORRUPTED LATEX and a
BUG DESCRIPTION that names exactly one intended bug (e.g., missing package,
wrong environment, label mismatch).
Set verdict=Pass only if you correctly detect the described bug (or correctly
report NO bug on clean controls).
Return ONLY JSON:
{ "verdict":"Pass|Fail", "error_types":[...], "notes":"" }
```

Contrast Judge (Blind)

```
You are a strict LaTeX bug detector. You will receive LATEX CODE that may
contain exactly ONE bug (missing package, wrong environment, label mismatch,
booktabs downgrade, illegal sectioning). Decide if a bug is present, and if so,
name the bug type from:
["Package Missing (graphicx)", "Wrong Env (figure->table)",
```

```
 "Illegal Sectioning (chapter in article)", "Label Mismatch",
 "Booktabs Downgrade"].
Return ONLY JSON:
{ "verdict":"Pass|Fail", "bug_present": true|false,
  "bug_type":"" or one of the above, "error_types":[...], "notes":"" }
```

E Output Contracts (JSON Schemas)

All judge outputs must conform to the following JSON Schemas (Draft-07). These schemas are used for validation prior to scoring.

Generation / Edit / Contrast (Described Bug)

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "LaTeX Judge Output",
  "type": "object",
  "additionalProperties": false,
  "required": ["verdict", "error_types", "notes"],
  "properties": {
    "verdict": { "type": "string", "enum": ["Pass", "Fail"] },
    "error_types": {
      "type": "array",
      "items": {
        "type": "string",
        "enum": ["Capability Error","Syntax Error","Logical Error",
                 "Package Error","Formatting Error","Constraint Violation"]
      }
    },
    "notes": { "type": "string" }
  }
}
```