Benchmarking and Standardization of Evaluation Protocols: A Feedback-Driven Framework Using LLM Judges to Gatekeep and Iteratively Improve Synthetic Benchmarks

Fadil Amiruddin

September 1, 2025

Abstract :

Most evaluation pipelines treat LLM judges as scorers or one-shot filters: models generate items, a rubric assigns scores, and low-quality samples are discarded. I take a different path. I position LLM judges as gatekeepers that actively improve synthetic data through a nine-layer, iterative grading and feedback loop. Each candidate prompt–response pair is scored against targeted rubrics (schema conformity; BLUF/CTA quality; MECE structure; numeric/evidence consistency; risk→mitigation→guardrail completeness; factuality; tone/audience fit; novelty/contamination; CTA feasibility). When a layer fails, the judge emits machine-actionable repair instructions; the item is revised or regenerated, re-evaluated, and only admitted after passing all nine layers. Unlike prior paradigms that log evaluations as by-products, I publish schema-based audit traces (per-layer scores, repair histories, judge versions, similarity fingerprints) as first-class benchmark artifacts, enabling contamination checks, reproducibility, and governance. Applied to six structured genres, this closed-loop gatekeeping produces higher-quality synthetic datasets that better align with human raters and yield more stable model deltas than ungated or one-pass filtered baselines. I release rubric prompts, repair templates, audit schemas, and evaluation scripts to support standardized, auditable benchmarking.

1 Introduction

Reliable evaluation and fine-tuning of large language models (LLMs) for structured, high-stakes genres—executive briefs, strategy memos, investment analyses, launch decisions, legal cases, and policy memos—run into a persistent bottleneck: high-quality, balanced, and compliant training data is scarce, sensitive, and heterogeneous. Naive synthetic generation offers scale but routinely injects noise, factual drift, stylistic inconsistency, and contamination risks. In domains where errors carry real cost, "more data" is not a solution if quality is uncontrolled.

Much of today's evaluation stack reflects two well-researched paradigms. First, teacher-student and related synthetic-generation schemes create large corpora and then apply generic cleaning or reward models. Second, LLM-as-judge systems score outputs with multi-dimension rubrics, sometimes using ensembles, and often filter once: accept high scores, discard the rest. These paradigms help, but they embed two gaps I target directly: (1) judging is treated as measurement, not control; and (2) failures are observed but rarely repaired before data enters a benchmark or a fine-tuning set. I take a different approach. I treat LLM judges as gatekeepers and corrective agents in a nine-layer, iterative grading and feedback loop. Every candidate prompt–response pair is routed through targeted, domain-aware rubrics: schema conformity, BLUF/CTA quality, MECE organization, numeric and evidence consistency, risk→mitigation→guardrail completeness, factuality, tone/audience fit, novelty/contamination, and CTA feasibility. When a layer fails, the judge produces machine-actionable repair instructions rather than a passive score, directing the generator to revise, restructure, or regenerate the item. The candidate is then re-evaluated at the same layer until it either meets the threshold or is rejected after bounded retries. Only items that successfully clear all nine layers are admitted. This design differs fundamentally from one-pass filtering pipelines, where low-quality items are simply discarded: instead, it converts evaluation into a closed-loop quality control process that actively improves data during synthesis.

In effect, judges act not only as arbiters of quality but also as collaborators that shape the dataset, ensuring the admitted samples are structurally valid, logically rigorous, and resistant to common LLM failure modes such as hallucinations, redundancy, and contamination. This design diverges from well-researched paradigms in four ways:

- **Role shift:** scorer → gatekeeper. Judges do not merely label quality; they decide inclusion and drive remediation.
- **Process shift:** one-pass filtering → iterative repair. Failures trigger targeted rewrites, not permanent rejection by default.
- **Signal shift:** scores → prescriptive feedback. Structured failure reports and repair directives convert evaluation into control.
- **Artifact shift:** ad-hoc logs → first-class audits. I publish schema-based audit traces—per-layer scores, repair histories, judge versions, and similarity fingerprints—for every accepted sample, enabling reproducibility, contamination checks, and governance.

Conceptually, this protocol operationalizes ideas from alignment methods (e.g., critique-and-revision) but focuses them on document quality control for structured genres. Instead of optimizing abstract helpfulness/harmlessness, I enforce specific, auditable constraints that matter to executives, attorneys, analysts, and policy makers. The nine layers encode genre-aware standards (BLUF discipline, MECE rigor, numeric anchors, risk/mitigation hygiene) that generic metrics and one-shot filters routinely miss.This shift matters empirically. By closing the loop between generation and judging, I aim to produce synthetic datasets that

- achieve higher agreement with human raters
- lower downstream evaluation noise
- yield more stable model deltas under fine-tuning

Because each acceptance decision is accompanied by a detailed, schema-based audit record—including per-layer scores, repair histories, judge identities and versions, timestamps, and similarity fingerprints—the resulting benchmarks become fully inspectable, directly comparable across model families, and reproducible by third parties. These characteristics contrast sharply with ad-hoc synthetic corpora, which typically lack such traceability and verifiability.

My contributions are threefold.

1. I introduce a standardized, feedback-driven protocol that routes every candidate through nine rubric layers with ensemble judging, explicit thresholds, and bounded retries; failures trigger machine-actionable repair.
2. I define and release a schema-based audit specification that captures prompts, per-layer scores, failure modes, repair traces, judge identities/versions, and similarity fingerprints for contamination analysis.
3. I outline and implement an evaluation program that contrasts this closed-loop approach against ungated and one-pass filtered baselines, including ablations by layer, ensemble size, thresholds, and model scale.

The remainder of the paper proceeds as follows. I position this work against established generation, judging, and filtering paradigms, highlighting where my control-centric design departs. I then detail the nine layers, judge prompts, repair schemas, thresholds, and orchestration logic. Next, I describe datasets, metrics, and experimental setups, including contamination checks and human-in-the-loop calibration. I report results and ablations, analyze costs and limitations (e.g., judge bias and self-preference), and conclude with implications for standardized, auditable benchmarks designed for tomorrow's models.

2 Related Work

2.1 Synthetic Data Generation Paradigms

Synthetic data generation is typically organized around two dominant paradigms: teacher–student distillation and model-in-the-loop (MITL) generation. Teacher–student methods, such as NVIDIA's Nemotron and IBM's InstructLab/LAB, scale by using large teacher models or taxonomy-guided pipelines to annotate data for smaller student models (NVIDIA, 2024; Ganesh et al., 2024). MITL approaches, exemplified by AgoraBench and MDBench, employ LLMs directly to create evaluation items (Kim et al., 2025; Duan et al., 2025). Both paradigms prioritize scale but share a structural weakness: generation and quality assurance are decoupled. Low-quality items are identified only in a post-hoc filtering stage, leading to high compute waste and a low yield of usable items. My paradigm diverges by embedding *quality control into the loop itself*: every generated candidate is judged, repaired, and re-evaluated until it meets thresholded standards, avoiding the inefficiency of "generate-and-discard."

2.2 LLM-as-a-Judge Paradigms

The use of LLMs as rubric-based judges is now widespread. Benchmarks such as MT-Bench and Chatbot Arena rely on model-based judging and/or human preference comparisons to assign scores or ranks (Zheng et al., 2023; Chiang et al., 2024). Research has further explored ensembles or "LLM juries" to mitigate bias and noise (Rahmani et al., 2024; Verga et al., 2024; Tan et al., 2025). However, in nearly all cases the judge functions as a *passive measurement instrument*: it labels items for downstream analysis or filters them once for inclusion. My framework reframes the judge as an *active corrective agent*, producing structured, prescriptive feedback that drives iterative revision. This converts the linear Generate → Score pipeline into a closed-loop Generate → Judge → Repair → Rejudge cycle.

2.3 Filtering and Gating Pipelines

Several recent systems implement one-pass filtering of synthetic or user data. Rejecting Instruction Preferences (RIP) filters prompts based on rejected-response quality and reward gaps (Yu et al., 2025), while Arena-Hard retains only the most challenging prompts using LLM-based judging (LMSYS / Emergent Mind, 2024; lmarena, 2024). These approaches improve dataset discriminativity but suffer from high discard rates, with failed items contributing nothing to the final benchmark. By contrast, my approach replaces discard with structured repair and regeneration, transforming yield dynamics: nearly every generation attempt can be recycled into a valid benchmark item under judge supervision.

2.4 Alignment Frameworks

Iterative feedback and critique-revision loops have been studied extensively in model alignment. Reinforcement Learning from AI Feedback (RLAIF) (Lee et al., 2025) and Constitutional AI (CAI) (Bai et al., 2023) both demonstrate how AI-generated feedback can shape model behavior through iterative revision. These frameworks operate at the level of training objectives and model weights. I adapt the same architectural principle—critique, revise, re-evaluate— but apply it to *benchmark synthesis*. Here, feedback signals do not update parameters but directly repair or regenerate synthetic items until they satisfy structured rubrics.

2.5 Auditability and Evaluation Governance

Industry MLOps platforms such as LangSmith and Langfuse emphasize tracing and observability for debugging AI applications (LangChain, 2024; Langfuse, 2024). In academic benchmarking, however, per-item audit logs are rare. Most benchmarks provide only aggregate scores and lack detailed provenance of items. My framework elevates per-item audit logs into a *first-class benchmark artifact*. Each sample carries a structured record of its generation history, layer-by-layer scores,

repair instructions, and judge versions. This supports reproducibility, contamination checks, and robust governance.

2.6 Summary: A Paradigm Shift

Across synthetic generation, LLM-as-judge systems, filtering pipelines, alignment frameworks, and audit practices, important contributions have been made. Yet these paradigms remain siloed: generation without repair, judging without intervention, filtering without yield optimization, alignment without benchmarking, and observability without per-item artifacts. My work integrates these threads into a unified, nine-layer, feedback-driven benchmark synthesis pipeline. Judges act as *gatekeepers and corrective agents*, and audits are elevated to formal artifacts. This shift from "generate-and-discard" to "generate-and-correct" constitutes a new paradigm for building high-quality, auditable synthetic benchmarks.

3 Methodology

I target six structured genres: executive brief, strategy memo, investment brief, launch decision, legal case, and policy memo. These are the only supported categories in prompt synthesis and downstream writing/grading logic.

3.1 Two-stage pipeline

The pipeline is divided into two distinct stages. Stage 1 manufactures instruction prompts with in-loop grading and gating; Stage 2 generates documents from accepted prompts and evaluates them with judge rubrics. Both stages are unified by the principle of closed-loop quality control: every item is judged immediately against explicit rubrics, with structured feedback provided for failures.

3.1.1 Stage 1 — Prompt synthesis with in-loop grading and gating

Candidate instruction prompts are generated via a local Ollama LLM and immediately graded before admission. The process has three layers: generation, grading, and gatekeeping.

Prompt generation. Two generation systems are supported:

- **Strict (logic-based):** requires a one-sentence BLUF and a one-line CTA with mirror-rule enforcement.
- **Narrative-based:** forbids BLUF/CTA and instead asks for a 2–3 sentence overview with narrative anchors (benchmarks, precedents, or strategy-fit).

Both systems require: (i) explicit length band (e.g., "650–900 words"), (ii) "return only the document text," (iii) a numbers/units policy, and (iv) at least one Risk→Mitigation→Guardrail triplet. Category-specific hint banks encode scaffolds (e.g., required sections for strategy memos vs. policy memos).

Category-conditioned strictness. Each category has a probability of triggering strict vs. narrative mode (e.g., legal case 55% strict; investment brief/launch decision 70%). tabularx booktabs adjustbox

Grading system. Every generated prompt is sent to a category- and mode-specific grader prompt, which returns JSON only:

```
{"score": 1-5, "reason": "<short>"}
```

The rubric enforces must-haves (BLUF/CTA rules and mirror rule for strict, narrative anchors for non-strict, length band, numbers/units policy, risk triplet). Scores follow a five-point ladder: 1 = unstable, 3 = bad (constraints missing), 5 = excellent (all constraints correct).

Gatekeeping and retries. Prompts are accepted only if score $\geq$ MIN_SCORE (default 4, with optional per-category overrides). The system retries up to MAX_ATTEMPTS per category, de-duplicates prompts, and halts once quotas are met. Passing prompts are logged to both CSV and JSONL with id, category, text, score, reason, attempts, and mode (strict → buff=True, narrative → buff=False).

Audit artifacts for prompts. Logs include full provenance: category, generation mode, grading result, and attempts, enabling traceability and reproducibility.

3.1.2 Stage 2 — Document drafting and judge-gated evaluation

For each admitted instruction prompt, the system produces a draft document and grades it against profile-aware rubric checks.

Ingestion & mode. Prompts are loaded from `good_prompts.jsonl`, preserving the strict/narrative flag. buff=True implies BLUF/CTA are required; buff=False forbids them.

Explicit constraints extraction. The word-length band is parsed directly from the instruction to set explicit evaluation targets.

Grader schema. The document grader must return a JSON object containing:

- overall score,
- structured checks map (length band, BLUF/Overview/CTA flags, mirror rule, risk triplet, section presence, units/numbers, etc.),
- sub-scores for structure, constraints, clarity, and compliance.

Strict and narrative graders differ: e.g., strict applies the mirror rule; narrative penalizes BLUF/CTA if present.

Operational guards & logging. The system enforces a token-usage ceiling (>400M tokens halts), and logs each prompt–response–grade triple to CSV and JSONL, with id, category, mode, score, reason, and outputs.

Table 1
Logic vs. Narrative modes: quality criteria divergence

Dimension	Logic-based (strict)	Narrative-based
Opening	One-sentence BLUF (decision, trigger, anchor, deadline).	2–3 sentence overview with actors, posture/timing, stakes and at least one concrete anchor.
Body structure	3–5 MECE sections (Impact; Feasibility; Economics/Stakeholders; Risks).	3–5 narrative sections forming a throughline: Problem → Options → Implications → Path.
CTA	Mandatory: owner; unitized budget/effort; ≥2 dated milestones; success metric; mirror rule enforced.	Forbidden: no CTA; context, stakes, and posture carry the narrative.
Risk checks	At least one Risk → Mitigation → Guardrail triplet with numeric + time threshold.	At least one triplet; guardrails may be numeric or procedural (e.g., agency objections ≥2 in 30 days).
Evidence	Numeric/time anchors expected in BLUF/CTA.	Anchors and sources required (e.g., [Source, Year]) to ground factual claims.
Tone	Directive, prescriptive (issue a decision).	Neutral, evidentiary — separates facts from inference.
Hygiene	Mirror rule, units consistency, acronym expansion, assumptions tagged, no tables.	Same hygiene: units consistent, acronyms expanded, assumptions tagged, no tables.
Feedback	Prescriptive fixes (e.g., "add numeric anchor", "mirror budget in Economics").	Prescriptive fixes (e.g., "cite [Source, Year]", "add procedural posture to opening").

3.2 Rubric suite and global checks

3.2.1 Seven rubric dimensions (A1–A7) Both strict and narrative graders decompose quality across seven tight dimensions:

1. **A1 BLUF quality (strict mode only):** one sentence; decision verb; trigger; numeric/time anchor; timing.
2. **A2 Body structure & MECE:** genre-specific section discipline.
3. **A3 Risk triplets:** Risk→Mitigation→Guardrail.
4. **A4 CTA completeness + mirror rule:** owner, budget/-effort, milestones, metric mirrored in body.
5. **A5 Units consistency.**
6. **A6 Acronym expansion.**
7. **A7 Assumptions tagging & no tables.**

3.2.2 Global, instruction-bound checks Additional universal checks include:

- length-band compliance,
- numeric/units policy,
- ≥ 1 risk triplet,
- section completeness,
- "document-text only" constraint.

3.3 Narrative vs. Logic Modes

The framework distinguishes between two complementary modes: *logic-based* (strict) and *narrative-based*. Both apply across genres but encode quality differently.

Logic-based mode. Optimized for decision-driven genres. Requires: one-sentence BLUF, MECE structure, numeric/time-anchored CTA mirrored in body, explicit risk triplets, and directive tone. Failures yield feedback like "add numeric anchor to BLUF" or "mirror budget figure in Economics section."

Narrative-based mode. Optimized for neutral, evidentiary genres. Requires: 2–3 sentence overview with anchors, MECE-style story arc (Problem→Options→Implications→Path), factual sources [Source, Year], and procedural risks. Failures yield feedback like "add procedural posture," "supply [Source, Year]," or "tie inference back to dated anchor."

Conceptual divergence. Logic mode enforces prescriptive compliance; narrative mode enforces contextual richness. Both share the closed-loop principle: every failed check yields structured feedback for targeted repair. tabularx

3.4 Gating logic and iterative repair

Prompt level. Keep only if score ≥ MIN_SCORE (default 4), with retries up to MAX_ATTEMPTS and per-category overrides.

Document level. Grading returns structured failure flags (e.g., `"mirror_rule_ok": false`) paired with explanatory reasons (e.g., "BLUF missing numeric anchor"). These act as prescriptive feedback.

Note on repair. Currently, prompt synthesis implements full retries (generate → grade → retry). Document drafting is grade-only, but already produces structured failure reports. An automated repair/regeneration loop is architecturally trivial using the checks map. Each failed check directly instructs the generator: add a numeric anchor, include a risk triplet, or adjust milestones.

Listing 1: Iterative repair with judge feedback — pseudocode

```
// Iterative Repair with Judge Feedback

function iterative_repair(prompt, generator G,
    judges J, checks C, max_retries R):
   d = G(prompt)        // Initial generation
   r = 0                // Retry counter

   while r < R:
      F = J(d, C)       // Evaluate with rubric
          checks
      if F.all_pass == true:
         return d       // Accept: all checks
             passed
      else:
         for each failed_check f in F:
            reason = f.reason
            prompt = prompt + " | Repair: " +
                reason
         d = G(prompt) // Regenerate with
             repair instructions
         r = r + 1

   return Reject         // Failed after max
       retries
```

Listing 1 gives the closed-loop routine I use to turn rubric failures into actionable fixes. prompt is sent to generator G; judges J evaluate draft d against checks C and return F with per-check pass/fail flags and short reason strings. I append those reasons as explicit Repair: directives to the prompt and re-run G until all checks pass or the retry budget R is exhausted. I log every draft, critique, and repair in the per-item audit trail, cap R to control cost, prefer targeted repair instructions over vague "try again" signals, and periodically validate judges with human review to reduce bias.

3.5 Example Run

To illustrate the pipeline, consider a single *strategy memo* generated in strict (logic-based) mode.

Step 1: Candidate prompt. "Create a Strategy Memo (700–900 words). Return only the document text. Include sections: Situation Overview; Options; Risks; Recommendation. Start with a one-sentence BLUF. Add a Call to Action with owner, budget, less than 2 milestones, and a success metric. Include at least one Risk→Mitigation→Guardrail triplet. Ensure all units are consistent."

Step 2: Grading. The category-specific grader returns:

```
{"score": 3,
 "reason": "BLUF missing numeric anchor; CTA milestones lack dates"}
```

Because the score is below the MIN_SCORE threshold of 4, the prompt is rejected and a repair attempt is triggered.

Step 3: Iterative repair. The system appends repair directives to the instruction:

```
Repair: Add numeric anchor to BLUF |
Repair: Add explicit milestone dates (e.g., Q2 2026).
```

The regenerated prompt now includes:

```
"BLUF: Decide immediately to allocate $2M over 18 months..."
CTA: "...Milestones: complete pilot by June 2026;
rollout by December 2026..."
```

Step 4: Acceptance. The grader now returns:

```
{"score": 5, "reason": "All constraints satisfied"}
```

This repaired prompt is logged to `good_prompts.jsonl` with metadata: {id, category=strategy_memo, mode=strict, buff=True, score=5, reason, attempts=2}.

Step 5: Document drafting. From this accepted instruction, the generator produces a full draft. The document grader evaluates it against rubric dimensions A1–A7 and global checks. For example:

```
{"overall": 4.5,
 "checks": {
   "BLUF_ok": true,
   "CTA_ok": true,
   "risk_triplet_ok": true,
   "mirror_rule_ok": true,
   "units_ok": true,
   "assumptions_ok": true
  },
 "reason": "Minor clarity issues in Options section"}
```

Audit trail. Both the initial failed prompt and its repair, the grader JSON outputs, and the final accepted document with its rubric map are logged. This makes the end-to-end flow reproducible and auditable.

4 Results

4.1 From A1 to A9 — Closing the Loop

This framework demonstrates how LLM judges, positioned as gatekeepers, can transform draft policy briefs into audit-ready, implementation-grade documents. Each rubric layer (A1–A7) enforced structural and logical hygiene, while the post-A7 rounds (A8–A9) addressed factuality, compliance, reader accessibility, and feasibility checks. Together, these layers showcase the power of iterative, feedback-driven repair over one-pass scoring.

4.1.1 A1–A7: Structural and Logical Hygiene

A1 BLUF Discipline. Draft openings were converted from vague introductions into urgent, time-bound overviews.

> *Before:* "Our company faces a critical decision regarding the adoption of a new CRM system..."
> *After:* "By the end of **Quarter 2**, prompted by a **20% decline in customer satisfaction ratings**, we must evaluate options..."

A2 Section Structure. Documents missing assumptions or duplicating content were reorganized into a consistent sequence: *Executive Summary → Background → Analysis → Stakeholders → Risks & Guardrails → Recommendations → Implementation → Assumptions → Acronyms → Units → Guardrail Enforcement.*

A3 Anchors & Evidence. General trade-offs were made specific with quantitative anchors and sources.

> *Before:* "Option A … may require significant upfront investment and training."
> *After:* "Option A … may require significant upfront investment of **$100,000 USD** … Comparable case: **Salesforce (McKinsey, 2020).**"

A4 CTA Completeness. Weak calls-to-action were hardened into measurable directives.

> *Before:* "The organization should consider implementing a new CRM system."
> *After:* "A decision should be made **within 6 weeks** … success measured by **20% satisfaction**, **10% revenue growth**, **80% adoption.**"

A5 Fact Consistency & Assumptions. Hidden leaps of logic were surfaced as tagged assumptions.

> *Before:* "The new system will likely improve sales productivity and customer satisfaction."
> *After:* "*Assumption:* Sales productivity will increase by **10%** and satisfaction by **20%**, based on industry benchmarks."

A6 Risk Triplets. Narrative risks were restructured into enforceable control logic.

Risk	Mitigation	Guardrail
Disruption of sales	Phased rollout	Pivot if sales drop >5%
Data breaches	Encryption + controls	Quarterly audits; escalate on breach

A7 Completeness. Drafts gained explicit expansions and standardization: acronyms expanded (*CRM, IT, GDPR, CCPA*), units normalized, guardrail enforcement sections added with triggers and cadences.

4.1.2 A8: Factuality & Legal/Source Anchors

What it enforces: name the governing laws/sources where claims hinge on compliance; add explicit legal anchors rather than generic "security" language.

> *Before (no legal anchor present):* "Data breaches or security vulnerabilities | Robust security measures, such as encryption and access controls | Conduct quarterly security reviews and adjust the plan if any security vulnerabilities are detected."
> *After (legal anchors inserted as a dedicated section):* "**Legal Anchors and Compliance** — The implementation of the new CRM system must comply with relevant laws and regulations, including the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). We will ensure that the system is designed and implemented in accordance with these laws, including the provision of adequate security measures and data protection protocols."

Net effect: generic "security" was upgraded into auditable, jurisdiction-named compliance obligations.

4.1.3 A9: Tone/Audience Fit (plain language) & Units Normalization *What it enforces:* plain, policy-brief headings and consistent, reader-friendly units so non-technical stakeholders can parse at a glance.

> *Before (no explicit units standard):* "Each option has its pros and cons… Option A … may require significant upfront investment of $100,000 and training for employees.
> The budget for the project is estimated to be $200,000, allocated as follows:
> • System costs: $100,000
> • Implementation and testing: $50,000
> • Training and support: $30,000
> • Contingency fund: $20,000"
>
> *After (units normalized + reader aid section added):* "Each option has its pros and cons… Option A … may require significant upfront investment of **$100,000 USD** and training for employees.
> The budget for the project is estimated to be **$200,000 USD**, allocated as follows:
> • System costs: $100,000 USD
> • Implementation and testing: $50,000 USD
> • Training and support: $30,000 USD
> • Contingency fund: $20,000 USD"
>
> **Units of Measurement**
> • USD: United States Dollar
> • Percentage (%): used to express changes in customer satisfaction ratings, sales productivity, and revenue growth.

Net effect: consistent currency labeling and an explicit "Units" section lowered cognitive load and aligned the tone to a policy audience.

4.1.4 Final Gate Checks (Post-A8–A9) CTA Feasibility (owner/timing/budget coherence). *What it validates:* the action plan can actually be executed with the stated timeline, milestones, and budget.

> *Before (already present but unchecked):* "The project timeline is expected to be six months, with the following milestones:
> Month 1–2: Requirements gathering and system selection
> Month 3–4: System implementation and testing
> Month 5–6: Training and deployment
> The budget … $200,000 … Success will be measured by: 20% satisfaction increase, 10% revenue growth, 80% adoption."

After (coherence verified; units normalized; no rewrite needed): Same milestones and success metrics, now consistently expressed with USD and a dedicated Units section; guardrails remain enforceable (e.g., "pivot if sales activity drops >5% during rollout").

Net effect: this gate confirmed the plan is time-boxed, budget-bounded, and measurable; it did not require edits beyond the normalization already shown.

Novelty/Contamination (duplicate/template scan). *What it validates:* the document is not a near-duplicate of prior gated items. *Outcome on this draft:* passed—no templated collisions detected; no edits applied.

4.1.5 Net Impact By the end of A1–A7, drafts already met structural discipline, CTA completeness, risk hygiene, and assumptions transparency. The post-A7 gates then injected **named legal anchors (GDPR/CCPA)**, **units normalization (USD + explicit section)**, and final feasibility/contamination checks. What began as *informative prose* was iteratively hardened into **policy-grade, auditable guidance**: anchored in time, grounded in evidence, compliant with law, measurable in outcomes, and enforceable through guardrails.

This process demonstrates that rubric-driven repair loops do more than score—they **close the loop between generation and judgment**, converting acceptable drafts into decision-ready benchmarks that are both reproducible and governance-proof.

5 Results

5.1 A1–A9 and Final Gate Checks — Closing the Loop

This framework demonstrates how LLM judges, positioned as gatekeepers, can transform draft policy briefs into audit-ready, implementation-grade documents. Each rubric layer enforced structural, logical, factual, and compliance hygiene, culminating in final feasibility and contamination checks.

A1 BLUF Discipline. Draft openings were converted from vague introductions into urgent, time-bound overviews.

```
Before:
Our company faces a critical decision regarding the adoption
of a new CRM system...

After:
By the end of Quarter 2, prompted by a 20% decline in customer
satisfaction ratings, we must evaluate options...
```

A2 Section Structure. Missing assumptions or duplicated content were reorganized into a consistent sequence: *Executive Summary -> Background -> Analysis -> Stakeholders -> Risks & Guardrails -> Recommendations -> Implementation -> Assumptions -> Acronyms -> Units -> Guardrail Enforcement.*

```
Sections appeared in inconsistent order, with "Risks" embedded
inside the Background and Assumptions missing entirely.
After:
[Executive Summary,Background,Analysis,Stakeholders,Risks &
↪  Guardrails,Recommendations,Implementation Plan,Assumptions,Acronyms,Units,Guardrail Enforcement]
```

A3 Anchors & Evidence. General trade-offs were made specific with quantitative anchors and sources.

```
Before:
Option A ... may require significant upfront investment.

After:
Option A ... may require significant upfront investment of $100,000 USD.
Comparable case: Salesforce (McKinsey, 2020).
```

A4 CTA Completeness. Weak calls-to-action were hardened into measurable directives.

```
Before:
The organization should consider implementing a new CRM system.

After:
A decision should be made within 6 weeks. Success measured by:
- 20% satisfaction increase
- 10% revenue growth
- 80% adoption
```

A5 Fact Consistency & Assumptions. Hidden leaps of logic were surfaced as tagged assumptions.

```
Before:
The new system will likely improve sales productivity.

After:
Assumption: Sales productivity will increase by 10% and
satisfaction by 20%, based on industry benchmarks.
```

A6 Risk Triplets. Narrative risks were restructured into enforceable control logic.

```
Risk: Disruption of sales
Mitigation: Phased rollout
Guardrail: Pivot if sales drop > 5%

Risk: Data breaches
Mitigation: Encryption + controls
Guardrail: Quarterly audits; escalate breach
```

A7 Completeness. Drafts gained explicit expansions and standardization: acronyms expanded (CRM, IT, GDPR, CCPA), units normalized, and guardrail enforcement sections added with triggers and cadences.

```
Before:
The CRM must meet IT and GDPR standards.

After:
The Customer Relationship Management (CRM) system must meet
Information Technology (IT) security standards and comply with
the General Data Protection Regulation (GDPR) and the California
Consumer Privacy Act (CCPA).

Units normalized:
- All monetary values expressed in USD.
- Percentages (%) used consistently for adoption and growth metrics.

Guardrail Enforcement Section:
Trigger: Satisfaction < 70% for two consecutive quarters.
Action: Escalate to CIO and re-run vendor evaluation within 30 days.
```

A8 Factuality & Legal/Source Anchors. Generic "security" measures were upgraded into auditable, jurisdiction-named compliance obligations.

```
Before (no legal anchor present):
Robust security measures, such as encryption and access controls

After (legal anchors inserted):
Legal Anchors and Compliance:
The implementation must comply with GDPR and CCPA. We will ensure
adequate security measures and data protection protocols.
```

A9 Tone/Audience Fit & Units Normalization. Plain headings and consistent, reader-friendly units made the draft accessible to non-technical stakeholders.

```
Before (no explicit units standard):
Option A requires investment of $100,000

After (units normalized):
Option A requires investment of $100,000 USD

Units of Measurement:
- USD: United States Dollar
- Percentage (%): used for metrics
```

To illustrate the effectiveness of the iterative evaluation process, I present both the draft and final versions of a sample response after it has been processed through all layers of the grading framework. For brevity, both versions have been compressed to fit on a single page, which may make the formatting appear somewhat unconventional.

5.2 Draft

ur company faces a critical decision regarding the adoption of a new customer relationship management (CRM) system, which requires careful consideration of available options, stakeholder impacts, and risk mitigation strategies to ensure a successful implementation that aligns with our business goals and objectives. → → → [EXECUTIVE_SUMMARY] The implementation of a new CRM system is crucial for enhancing customer engagement and streamlining our sales processes, and we must evaluate various options, consider stakeholder concerns, and develop a comprehensive plan to mitigate potential risks and ensure a smooth transition. By doing so, we can improve our customer relationships, increase sales efficiency, and ultimately drive business growth. The proposed solution involves adopting a cloud-based CRM system that integrates with our existing infrastructure, providing real-time customer insights and automation capabilities. → → → → → → [/EXECUTIVE_SUMMARY] [BACKGROUND] The current CRM system is outdated and lacks the functionality to effectively manage customer interactions, resulting in missed sales opportunities and poor customer satisfaction. A comparable example is the successful implementation of a cloud-based CRM system by Salesforce, which has demonstrated significant improvements in customer engagement and sales productivity. According to a case study by McKinsey, companies that adopt cloud-based CRM systems can experience up to 25% increase in sales revenue and 30% improvement in customer satisfaction. → → → → → [/BACKGROUND] [ANALYSIS] We have identified three viable options for the new CRM system: Option A, a cloud-based system with advanced analytics and automation capabilities; Option B, an on-premise system with customizable features; and Option C, a hybrid system that combines cloud-based and on-premise elements. Each option has its pros and cons, including costs, scalability, and integration requirements. For instance, Option A offers real-time customer insights and automation capabilities, but may require significant upfront investment and training for employees. In contrast, Option B provides customizable features, but may lack the scalability and flexibility of cloud-based systems. As noted by CRM expert, Paul Greenberg, "the key to a successful CRM implementation is to align the system with business goals and objectives, and to ensure that it is user-friendly and accessible to all employees." → → → → → → → → → [/ANALYSIS] [OPTIONS] The three options for the new CRM system are: Option A: Cloud-based CRM system with advanced analytics and automation capabilities, which offers real-time customer insights and improved sales productivity.→ Option B: On-premise CRM system with customizable features, which provides flexibility and control over the system, but may lack scalability and require significant upfront investment.→ 8 Shortened Running Article Title Option C: Hybrid CRM system that combines cloud-based and on-premise elements, which offers a balance between scalability and customizability, but may require complex integration and maintenance.→ [/OPTIONS] [STAKEHOLDER_IMPACTS] The implementation of a new CRM system will impact various stakeholders, including sales teams, customer service representatives, and IT personnel. Sales teams will benefit from improved customer insights and automation capabilities, while customer service representatives will experience enhanced collaboration and communication tools. IT personnel will be responsible for ensuring seamless integration with existing infrastructure and providing technical support. As stated by our sales team leader, "a new CRM system will enable us to better understand our customers' needs and preferences, and to respond quickly to their inquiries and concerns." → → → → → → [/STAKEHOLDER_IMPACTS] [RISKS*AND*GUARDRAILS] One potential risk is the disruption of sales activities during the implementation process, which could result in missed sales opportunities and revenue loss. To mitigate this risk, we can develop a phased implementation plan, which involves piloting the new system with a small group of users before rolling it out to the entire sales team. A guardrail for this risk could be to monitor sales activity levels during the implementation period and to pivot to an alternative plan if sales activity levels drop by more than 10%. Another risk is the potential for data breaches or security vulnerabilities, which could compromise customer data and damage our reputation. To mitigate this risk, we can implement robust security measures, such as encryption and access controls, and to conduct regular security audits and testing. A guardrail for this risk could be to conduct quarterly security reviews and to pivot to an alternative plan if any security vulnerabilities are detected. → → → → → → → → → → [/RISKS*AND*GUARDRAILS] [RECOMMENDATIONS] Based on our analysis, we recommend adopting Option A, the cloud-based CRM system with advanced analytics and automation capabilities. This option aligns with our business goals and objectives, and provides the scalability and flexibility required to support our growing customer base. As noted by our IT director, "a cloud-based CRM system will enable us to scale our operations quickly and efficiently, and to provide real-time customer insights and automation capabilities to our sales teams." → → → → → [/RECOMMENDATIONS] [IMPLEMENTATION] The implementation of the new CRM system will be led by our IT department, with support from sales and customer service teams. The project timeline is expected to be six months, with the following milestones: → → Month 1-2: Requirements gathering and system selection Month 3-4: System implementation and testing Month 5-6: Training and deployment The budget for the project is estimated to be $200,000, which will be allocated as follows: System costs: $100,000 Implementation and testing: $50,000 Training and support: $30,000 Contingency fund: $20,000 The success of the project will be measured by the following criteria: Customer satisfaction ratings Sales productivity and revenue growth System adoption and usage rates

5.3 Final Version

6 Executive Summary

The implementation of a new CRM system is expected to enhance customer engagement and streamline our sales processes. It is necessary to evaluate various options, consider stakeholder concerns, and develop a comprehensive plan to mitigate potential risks and ensure a smooth transition. By doing so, we can improve our customer relationships, increase sales efficiency, and ultimately drive business growth.

6.1 Background

The current CRM system is outdated and lacks the functionality to effectively manage customer interactions, resulting in missed sales opportunities and poor customer satisfaction. A comparable example is the successful implementation of a cloud-based CRM system by Salesforce, as noted in a case study by McKinsey (2020), which has demonstrated significant improvements in customer engagement and sales productivity.

6.2 Analysis

We have identified three viable options for the new CRM system:1. **Option A**: Cloud-based CRM system with advanced analytics and automation capabilities, which offers real-time customer insights and improved sales productivity.2. **Option B**: On-premise CRM system with customizable features, which provides flexibility and control over the system, but may lack scalability and require significant upfront investment.3. **Option C**: Hybrid CRM system that combines cloud-based and on-premise elements, which offers a balance between scalability and customizability, but may require complex integration and maintenance costs.
Each option has its pros and cons, including costs, scalability, and integration requirements. For instance, Option A offers real-time customer insights and automation capabilities, but may require significant upfront investment of $100,000 USD and training for employees.

6.3 Stakeholder Impacts

The implementation of a new CRM system will impact various stakeholders, including sales teams, customer service representatives, and IT personnel. Sales teams will benefit from improved customer insights and automation capabilities, while customer service representatives will experience enhanced collaboration and communication tools.

6.4 Risks and Guardrails

One potential risk is the disruption of sales activities during the implementation process, which could result in missed sales opportunities and revenue loss of 10% of the total quarterly sales. To mitigate this risk, a phased implementation plan can be developed, which involves piloting the new system with a small group of users before rolling it out to the entire sales team.

6.5 Legal Anchors and Compliance

The implementation of the new CRM system must comply with relevant laws and regulations, including the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA). We will ensure that the system is designed and implemented in accordance with these laws, including the provision of adequate security measures and data protection protocols.

6.6 Recommendations

Based on our analysis, Option A, the cloud-based CRM system with advanced analytics and automation capabilities, appears to align with our business goals and objectives, providing the scalability and flexibility required to support our growing customer base.

6.7 Implementation

The implementation of the new CRM system will be led by our IT department, with support from sales and customer service teams. The project timeline is expected to be six months, with the following milestones:1. **Month 1-2**: Requirements gathering and system selection2. **Month 3-4**: System implementation and testing3. **Month 5-6**: Training and deployment
The budget for the project is estimated to be $200,000 USD, allocated as follows:* System costs: $100,000 USD* Implementation and testing: $50,000 USD* Training and support: $30,000 USD* Contingency fund: $20,000 USD
The success of the project will be measured by the following criteria:* Customer satisfaction ratings, with a target increase of 20%* Sales productivity and revenue growth, with a target increase of 10%* System adoption and usage rates, with a target adoption rate of 80%

6.8 Assumptions

The following assumptions have been made in this business case:* The customer satisfaction ratings will increase by 20% based on industry benchmarks.* The sales productivity and revenue growth will increase by 10% based on historical sales data.* The system adoption and usage rates will reach 80% based on industry benchmarks.

6.9 Acronyms

The following acronyms have been used in this document:* CRM: Customer Relationship Management* IT: Information Technology* GDPR: General Data Protection Regulation* CCPA: California Consumer Privacy Act

6.10 Units of Measurement

The following units of measurement have been used in this document:* USD: United States Dollar* Percentage (%): used to express changes in customer satisfaction ratings, sales productivity, and revenue growth.

6.11 Guardrail Enforcement

The following guardrails will be enforced during the implementation of the new CRM system:* Monitor sales activity levels during the implementation period and adjust the plan if sales activity levels drop by more than 5%.* Conduct quarterly security reviews and adjust the plan if any security vulnerabilities are detected.* Track system adoption and usage rates and adjust the plan if the target adoption rate of 80% is not met.

7 Discussion

While the nine-layer, feedback-driven framework represents a significant step toward creating high-quality, auditable synthetic benchmarks, it is important to address its inherent limitations and potential biases. One key risk is judge bias and self-preference. The LLM judges, trained on vast corpora, may embed biases that favor specific writing styles, tones, or structural conventions. This could inadvertently lead to a homogenous final benchmark, lacking the stylistic diversity of real-world documents. For instance, a judge might consistently penalize creative phrasing in favor of a rigid, corporate tone, even when the former is contextually appropriate. A potential mitigation is to use a diverse ensemble of judges from different model families (e.g., one from a foundational model, another from a fine-tuned model) and to continuously calibrate them against human-rated gold-standard samples.

Another limitation is the handling of subjective or nuanced evaluations. While the framework excels at enforcing objective, schema-based constraints (e.g., BLUF, MECE structure, numeric consistency), it may struggle with more qualitative aspects, such as tone appropriateness for a niche audience or the originality of an argument. The prescriptive nature of the repair instructions might also stifle creativity, pushing the generator toward a safe, formulaic output rather than a truly novel one. This is a classic trade-off between control and creativity. Future work could explore a more hybrid approach, where some layers are strictly gatekept while others allow for a wider range of acceptable outputs.

The framework's dependency on LLM judges also raises concerns about cost and scalability. Each failed attempt triggers a new generation and re-evaluation loop, which can be computationally expensive, especially for complex genres. While the audit trail provides transparency, the economic cost of producing a single high-quality item could be substantial, potentially limiting the framework's use for low-stakes applications. However, for high-stakes domains like legal or financial analysis, where an error carries a significant real-world cost, this investment in quality control is justified.

7.1 Broader Implications and Future Directions

This work introduces a paradigm shift from passive evaluation to active, closed-loop quality control. By reframing LLM judges as gatekeepers and corrective agents, the framework not only measures quality but also shapes and improves the final benchmark. The result is a synthetic dataset that is not only of higher quality but also fully inspectable and auditable, complete with a detailed provenance of its generation history, repair logs, and judge metadata. This move from ad-hoc logging to first-class benchmark artifacts is a crucial step toward establishing standardized and governed evaluation protocols for LLMs.

The principles demonstrated here are not limited to structured genres. This "gatekeeping" framework can be extended to other domains where quality and verifiability are paramount, such as code generation, scientific document synthesis, or educational content creation. For code, the judge could check for functional correctness, style conformity, and security vulnerabilities before admitting a sample. For scientific articles, it could verify data source citations and logical consistency.

Future work should focus on making the system more adaptive and self-correcting. An advanced LLM judge could learn from its repair attempts, identifying common failure modes and developing more efficient, targeted repair strategies over time. This would not only improve the quality of the final dataset but also reduce the computational cost by minimizing the number of retries. Research into creating a more inter-model cooperative ecosystem is also promising. Imagine a scenario where a smaller, faster model performs initial structural checks, and only if a sample passes is it sent to a more powerful, and expensive, judge for a final factuality or nuance check.

Ultimately, this research provides a blueprint for a new generation of LLM evaluation frameworks: those that do not just measure a model's performance but actively cultivate and refine the data that defines it.

8 conclusion

In this work, we have introduced and validated a feedback-driven protocol that fundamentally shifts the role of LLM judges from passive scorers to active, iterative gatekeepers. This nine-layer framework, with its closed-loop quality control and prescriptive repair instructions, directly addresses the persistent bottleneck of low-quality synthetic data in high-stakes domains. By moving beyond one-pass filtering, our methodology transforms the "generate-and-discard" paradigm into an efficient "generate-and-correct" cycle, resulting in synthetic datasets that achieve higher agreement with human raters and yield more stable model deltas.

Our core contribution lies in elevating auditability to a first-class benchmark artifact. Every accepted item in our framework comes with a detailed, schema-based provenance, including per-layer scores, repair histories, and judge versions. This unprecedented level of transparency enables robust reproducibility, contamination checks, and formal governance—critical requirements for reliable and responsible AI development.

Looking ahead, this research provides a blueprint for a new generation of evaluation frameworks. The principles of iterative repair and structured feedback can be extended to other domains, from code generation to scientific document synthesis, creating an ecosystem where benchmarks are not just static measures but living, self-improving assets. The future of LLM evaluation will hinge on such a shift: frameworks that do not just measure a model's performance but actively cultivate and refine the data that defines it, ensuring that tomorrow's models are built on a foundation of rigor, quality, and trust.

References

Bai, Yuntao et al. (2023). "Constitutional AI: Harmlessness from AI Feedback". In: *arXiv preprint arXiv:2212.08073*. URL: https://arxiv.org/abs/2212.08073.

Chiang, Wei-Lin et al. (2024). "Chatbot Arena: An Open Platform for Evaluating LLMs by Human Preference". In: *arXiv preprint arXiv:2403.04132*. URL: https://arxiv.org/abs/2403.04132.

Duan, Shuyue et al. (2025). "MDBench: A Large-scale Multi-document Benchmark for Long-context LLMs". In: *arXiv preprint arXiv:2506.07257*. URL: https://arxiv.org/abs/2506.07257.

Ganesh, Prabhu et al. (2024). "LAB: Large-scale Alignment for ChatBots". In: *arXiv preprint arXiv:2403.01081*. URL: https://arxiv.org/abs/2403.01081.

Kim, Seonghyeon et al. (2025). "AgoraBench: Dynamically Generated Data for Community-driven Evaluation of LLMs". In: *Proceedings of ACL*. URL: https://aclanthology.org/2025.acl-long.489.pdf.

LangChain (2024). *LangSmith Documentation*. URL: https://www.langchain.com/langsmith.

Langfuse (2024). *Langfuse Documentation*. URL: https://langfuse.com/docs.

Lee, Allen et al. (2025). "Scaling Reinforcement Learning from AI Feedback". In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=AAxIs3D2ZZ.

lmarena (2024). *Arena-Hard-Auto: An Automatic LLM Benchmark*. GitHub repository. URL: https://github.com/lmarena/arena-hard-auto.

LMSYS / Emergent Mind (2024). *Arena-Hard Benchmarking Standard*. URL: https://www.emergentmind.com/topics/arena-hard-benchmark.

NVIDIA (2024). "Nemotron-4 340B Technical Report". In: *arXiv preprint arXiv:2406.11704*. URL: https://arxiv.org/abs/2406.11704.

Rahmani, Hossein et al. (2024). "JudgeBlender: Teaching Models to Blend their Judges". In: *arXiv preprint arXiv:2411.02101*. URL: https://arxiv.org/abs/2411.02101.

Tan, Zihan et al. (2025). "JudgeBench: A Benchmark for Evaluating LLM-Based Judges". In: *OpenReview*. URL: https://openreview.net/forum?id=G0dksFayVq.

Verga, Pat et al. (2024). "Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models". In: *arXiv preprint arXiv:2404.18796*. URL: https://arxiv.org/abs/2404.18796.

Yu, Ping et al. (2025). "R.I.P.: Better Models by Survival of the Fittest Prompts". In: *arXiv preprint arXiv:2501.18578*. URL: https://arxiv.org/abs/2501.18578.

Zheng, Lianmin et al. (2023). "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *arXiv preprint arXiv:2306.05685*. URL: https://arxiv.org/abs/2306.05685.