Defense in Depth for Language-Model Applications: A Practitioner's Taxonomy of Evaluation, Retrieval Quality, Guardrails, and Hallucination Mitigation

Yash Singh
IIIT Una

18 July 2026

Abstract

Large language models (LLMs) fail in production for reasons that no single technique fully addresses: they fabricate facts, drift from retrieved evidence, ignore instructions, and answer confidently when they should abstain. The research literature offers a large and fast-moving catalogue of remedies, but practitioners face these methods as a disconnected toolbox rather than a coherent engineering discipline. This paper is a practitioner-oriented synthesis that organizes reliability techniques for LLM applications into four composable layers: (i) *evaluation*, the measurement substrate that turns reliability from a vibe into a number; (ii) *retrieval quality*, which governs whether a model is even given the evidence it needs; (iii) *inference-time reliability*, self-checking and abstention that catch errors before they leave the model; and (iv) *guardrails*, the programmable input/output controls that bound behavior. We map common failure modes to the layers that mitigate them, contrast the dominant methods on cost, latency, and the failures they actually catch, and propose a reference architecture that composes the layers as defense in depth. Our contribution is synthesis and framing rather than new empirical results: we argue that reliability is achieved not by any one method but by stacking cheap, imperfect filters, where distinct filters extend coverage across failure classes and redundant, independent filters drive the residual of any single class down multiplicatively. We close with the open problems that this framing exposes, chiefly evaluator trust, calibrated abstention, and the absence of end-to-end reliability budgets.

1 Introduction

A demo that works once and a product that works reliably are separated by a wide gap that LLMs make unusually treacherous. A retrieval-augmented assistant can answer a hundred questions correctly and then, on the hundred-and-first, confidently cite a policy that does not exist. The failure is not a crash with a stack trace; it is fluent, plausible, and invisible until a user is harmed. This is the central engineering problem of applied LLM work: the failure surface is open-ended and the failures are silent.

The research community has responded with a rich catalogue of methods: retrieval augmentation to ground generation in evidence [14, 23], self-consistency and verification to reduce reasoning errors [4, 24], black-box hallucination detectors [17], fine-grained factuality metrics [18], automated evaluation frameworks [5, 22, 26], and programmable guardrails [3, 20]. Each is valuable, but each is typically presented in isolation, benchmarked against a narrow slice of failures. A practitioner

who has read all of these papers still lacks the thing they most need: a way to reason about *how the pieces fit together* into a system whose overall failure rate is acceptable.

This paper offers that framing. We make three claims.

(1) Reliability is a stack, not a technique. No single method drives production failure rates low enough. Grounding reduces but does not eliminate hallucination; self-checking catches some but not all fabrications; guardrails block known-bad patterns but not novel ones. The path to reliability is defense in depth: multiple cheap, imperfect filters arranged so that residual error decays roughly multiplicatively rather than additively.

(2) The layers are mostly separable, and each answers a distinct question. We organize the toolbox into four layers, summarized in Figure 1: evaluation (*how wrong are we?*), retrieval quality (*did the model get the evidence?*), inference-time reliability (*does the model's own output survive scrutiny?*), and guardrails (*is the output within allowed bounds?*). Failure modes map to layers (Table 1), which lets a team locate the cheapest layer that addresses a given class of failure. The separation is a modeling convenience, not a hard boundary: adaptive methods like Self-RAG [2] deliberately fuse retrieval and self-critique into one loop, and we name such exceptions where they arise. The blur is benign because the taxonomy's job is to route a failure to the *cheapest* layer that catches it, not to forbid a method from spanning two.

(3) Evaluation is the load-bearing layer. Every other layer is only as trustworthy as the measurement that validates it. Because human labels do not scale, teams increasingly rely on LLM-as-judge and reference-free metrics [5, 26], which import their own reliability problem. We treat evaluator trust as the discipline's central open question.

Our aim is a map, not a new algorithm, and we are explicit about what is new and what is borrowed. The genuine unit of novelty is the *cross-family composition*: existing surveys catalogue methods *within* a single failure family, hallucination [9, 11] or retrieval [8], whereas we treat the whole pipeline as the object of analysis and ask how methods from different families compose into one system with a single, budgetable failure rate. The taxonomy, the failure-to-layer mapping, and the reference architecture are the scaffolding for that argument rather than novel artifacts in themselves; "defense in depth" is a borrowed security metaphor that we make quantitative through the residual-error and budget models of Section 8. The payoff is practical: a working engineer can use the composition to decide what to build first and where the residual risk lives.

2 Related Work

Our closest neighbors are surveys, and the distinction is deliberate. The hallucination surveys of Ji et al. [11] and Huang et al. [9], and the RAG survey of Gao et al. [8], are exhaustive *within* their family but stop at the family boundary: they taxonomize causes and list mitigations without asking how a retrieval fix and a guardrail trade off against each other under one shared error budget. Evaluation surveys and frameworks [5, 15, 22] similarly treat measurement as an end in itself rather than as the instrument that tunes every other layer. What none of these provide, and what we contribute, is the *cross-family* view: a single request path in which methods from different families are composed, and a residual-error model (Section 8) that makes their interaction budgetable.

The individual mechanisms we cite are, of course, prior work: retrieval grounding [14], self-checking and verification [4, 17, 24], uncertainty and abstention [1, 12, 13], programmable and learned guardrails [3, 10, 20], and attribution [7]. Our claim is not to improve any of them but to supply the missing arithmetic of how they stack.

Failure mode	Where it originates	Cheapest mitigating layer	Detectable?
Intrinsic hallucination	Generator drifts from context	Inference-time faithfulness check; RAG eval [5]	Yes (context present)
Extrinsic hallucination	Parametric knowledge is wrong	Retrieval grounding [14]; self-check [17]	Partially
Retrieval failure	Retriever recall / index	Retrieval quality layer; Self-RAG [2]	Yes (offline eval)
Instruction / format	Decoding ignores constraint	Output guardrail (schema/validator)	Yes (deterministic)
Safety / policy	Adversarial or unlucky input	Input+output guardrails [20]; RLAIF [3]	Partially
Miscalibration	No abstention signal	Confidence + abstention [12]	Hard

Table 1: Failure modes mapped to the layer that most cheaply mitigates them. "Detectable?" asks whether the failure can be caught automatically at acceptable cost, which determines whether it can be gated in a pipeline.

3 A Taxonomy of Failure

Reliability is undefined until failure is. We adopt a coarse but operationally useful taxonomy, aligned with the hallucination surveys of Ji et al. [11] and Huang et al. [9] but extended to the application concerns that dominate production.

Intrinsic hallucination. The output contradicts the provided source or context. In a RAG system this is a *faithfulness* failure: the retrieved passage says one thing, the answer says another. It is the most tractable failure because the ground truth (the context) is present at inference time.

Extrinsic hallucination. The output asserts facts that cannot be verified against any provided source and happen to be false. Closed-book factual errors live here. These are harder because verification requires knowledge the system does not hold.

Retrieval failure. The correct evidence was never surfaced: wrong chunk, poor recall, stale index, or a query the retriever could not match. No downstream layer can fix an answer built on absent evidence.

Instruction and format violation. The output ignores a constraint: wrong schema, disallowed content, exceeded length, broken tool call. These are, in many deployments, among the highest-frequency production failures and, fortunately, the easiest to detect deterministically.

Safety and policy violation. The output is harmful, discloses private data, or is successfully jailbroken [6, 19].

Miscalibration. The output is wrong *and* confident, or the system answers when it should have abstained. Calibration cuts across the other categories and determines whether the system knows what it does not know [12].

Table 1 maps each failure to the layers that most cheaply mitigate it. The recurring lesson is that failures are caught most cheaply close to their source: retrieval failures at the retrieval layer, format violations at the guardrail layer, and only the residue by expensive inference-time self-checking.

Approach	Measures	Strength	Limitation
Static benchmark [15, 16]	Model, in the abstract	Comparable, reproducible	Not your distribution
Behavioral tests [21]	Invariants on your system	Deterministic, CI-friendly	Only what you thought to assert
LLM-as-judge [26]	Subjective quality at scale	Cheap, flexible	Judge bias; needs calibration
RAG-specific [5, 22]	Faithfulness, relevance	Decomposed, actionable	Depends on judge quality
Atomic factuality [18]	Per-fact precision	Fine-grained	Costly; needs a knowledge source

Table 2: Evaluation approaches. No single row is sufficient; teams combine static benchmarks for model selection, behavioral tests in CI, and reference-free judges for continuous monitoring.

4 Layer 1: Evaluation

Evaluation is placed first not because it runs first at inference time but because it is the precondition for every other layer: a mitigation you cannot measure is a mitigation you cannot trust or tune.

Static benchmarks such as HELM [15] and TruthfulQA [16] give comparable, reproducible scores across models but answer a different question than production teams ask. They measure a model in the abstract; they do not measure *your* application on *your* distribution. They are necessary for model selection and nearly useless for regression-testing a specific pipeline.

Behavioral and unit-style testing, in the spirit of CheckList [21], closes part of that gap by asserting invariants (paraphrase invariance, directional expectations, known-answer probes) on the actual system. This is the LLM analogue of the unit test and belongs in CI.

Reference-free and LLM-as-judge evaluation has become the practical default because gold labels do not scale. MT-Bench and Chatbot Arena [26] popularized using a strong model as grader; RAGAS [5] and ARES [22] specialize this for retrieval systems, decomposing quality into faithfulness (is the answer grounded in retrieved context?), answer relevance, and context relevance. FActScore [18] pushes toward fine-grained factual precision by decomposing a generation into atomic facts and verifying each.

The catch is that an LLM judge is itself an LLM subject to every failure in Section 3: it has position and verbosity biases, it can be gamed, and it inherits the blind spots of its training. The practitioner's mitigation is to treat the judge as a measurement instrument that must itself be calibrated against a small human-labeled set, to prefer decomposed rubric scoring over holistic 1–10 ratings, and to track judge–human agreement as a first-class metric. Evaluation does not remove the reliability problem; it relocates it to a place where it can be sampled cheaply.

5 Layer 2: Retrieval Quality

Retrieval augmentation is the highest-leverage reliability intervention because it attacks extrinsic hallucination at the root: a model that is handed the answer is far less likely to invent one [8, 14, 23]. But RAG relocates rather than removes the problem. It introduces a new failure mode, retrieval failure, and a new dependency, the faithfulness of the generator to what it retrieved.

Three sub-problems dominate in practice. *Recall*: did the relevant chunk enter the candidate set at all? This is governed by chunking strategy, embedding quality, and hybrid (lexical plus dense) retrieval, and it is bounded above by the index. No re-ranker recovers a document that was never indexed. *Precision and ordering*: is the relevant chunk near the top, given finite context budget? Re-ranking addresses this. *Grounding*: given good context, does the generator actually use it? This is measured by faithfulness metrics [5] and is where intrinsic hallucination hides.

Adaptive schemes such as Self-RAG [2] let the model decide *when* to retrieve and *whether* retrieved passages are relevant, emitting reflection tokens that critique its own use of evidence. This begins to blur Layer 2 into Layer 3: retrieval quality and self-critique become a single loop. The practitioner's takeaway is that retrieval must be evaluated as its own component, with its own metrics (recall@k, context relevance) measured independently of end-to-end answer quality, because a good final answer can mask a broken retriever that the model happened to route around, and a bad answer can hide a perfect retriever undone by a generator that ignored it.

6 Layer 3: Inference-Time Reliability

The third layer operates on the model's own output at generation time, without external knowledge, exploiting a useful asymmetry: verifying a claim is often easier than producing it, and sampling the model multiple times exposes its uncertainty.

Self-consistency [24] samples several reasoning paths and takes the majority answer, trading compute for accuracy on tasks with a checkable final answer. **Chain-of-Verification** [4] has the model draft an answer, generate verification questions, answer them independently, and revise, which measurably cuts fabrication in list and entity tasks. **SelfCheckGPT** [17] exploits the observation that fabricated content is unstable across samples: if independently sampled responses disagree on a fact, that fact is likely hallucinated, giving a black-box detector that needs no external database.

Cutting across these is **calibration and abstention**. Models carry usable signal about their own correctness [12]; the reliability question is whether the system converts that signal into a decision to abstain, defer to a human, or hedge. Two threads sharpen the raw signal. *Semantic entropy* [13] measures uncertainty at the level of meaning rather than tokens, clustering sampled generations by semantic equivalence so that a model that says the same thing five different ways is correctly scored as confident, while one that contradicts itself is flagged; this makes the SelfCheckGPT intuition quantitative. *Conformal prediction* [1] goes further, wrapping any black-box scorer in a distribution-free procedure that returns prediction sets (or an abstention) with a finite-sample coverage guarantee, turning an ad hoc threshold into one with a provable error rate. In many production settings a calibrated "I don't know" is worth more than a marginally more accurate guess, because it converts a silent failure into a visible, routable one.

The cost of this layer is real: self-consistency and verification multiply token usage and latency. This makes Layer 3 the natural place for a *selective* policy, spending extra compute only on inputs the cheaper layers flag as uncertain or high-stakes.

7 Layer 4: Guardrails

Guardrails are the deterministic and semi-deterministic controls that bound what enters and leaves the model. They are conceptually the outermost shell in Figure 1 because they need no under-

standing of correctness, only of *allowed shape*.

Output validation is the cheapest and most reliable guardrail: schema/JSON validation, regex and type checks, tool-argument validation, and retry-on-invalid. Because format violations are deterministically detectable (Table 1), this layer can drive that failure class to near zero, and no probabilistic method should be used where a validator suffices.

Programmable rails such as NeMo Guardrails [20] generalize this to semantic policies: topical boundaries, canonical-form dialogue flows, and fact-checking rails, expressed in a controlled language that sits outside the model. **Learned classifier rails**, exemplified by Llama Guard [10], complement rule-based policies with a purpose-trained model that classifies both prompts and responses against an explicit safety taxonomy, catching the semantic violations a regex cannot express while remaining a separate, auditable component rather than a property of the generator. **Input rails** handle jailbreak and injection detection before the model is invoked, informed by red-teaming that systematically discovers failure inputs [6, 19].

Attribution rails sit between guardrails and evaluation: requiring the model to emit inline citations to its retrieved sources, and rejecting or regenerating any claim whose citation does not support it, converts an unverifiable assertion into a checkable one [7]. Attribution is a particularly cheap output rail in a RAG system because the evidence is already in context, and it directly attacks the intrinsic-hallucination class of Section 3.

Value alignment via feedback, exemplified by Constitutional AI [3], is a guardrail baked into the weights rather than bolted on around them: the model is trained to critique and revise its own outputs against an explicit set of principles. This complements rather than replaces external rails, since a trained-in disposition is cheaper at inference but harder to audit, update, and prove than an external rule.

The design principle for this layer is that guardrails should be composed by *cost and determinism*: run the cheapest deterministic check first, and reserve model-based rails for what deterministic checks cannot express.

8 A Reference Architecture

Figure 1 composes the four layers into a single request path. The arrangement encodes the paper's thesis: filters are ordered so that each removes the failures it catches most cheaply, and only the residue reaches the next, more expensive filter. The layer *numbers* follow exposition order rather than request order: the guardrail layer (Section 7) appears at *both* ends of the path, as input rails at the top and output rails at the bottom, because the same deterministic machinery bounds both the request and the response.

Two properties of this composition matter more than any single box.

Two mechanisms, not one. It is tempting to summarize defense in depth as "errors decay multiplicatively," but that slogan conflates two distinct effects, and only one of them is actually multiplicative. *Coverage* is a union effect: distinct layers catch distinct failure classes, so adding a layer that handles a previously unhandled class simply removes that class from the residual. Coverage is what the failure-to-layer map (Table 1) buys, and it is additive in the classes it eliminates, not multiplicative. *Redundancy* is the multiplicative effect, and it applies only *within* a class: if m independent filters each miss a fraction $\epsilon_1, \ldots, \epsilon_m$ of the *same* failure instances, the fraction surviving all of them is $\prod_i \epsilon_i$. Consider one high-stakes class guarded by three redundant, mechanistically different filters, each missing a fraction $\epsilon = 0.3$ of instances (these figures are pedagogical,

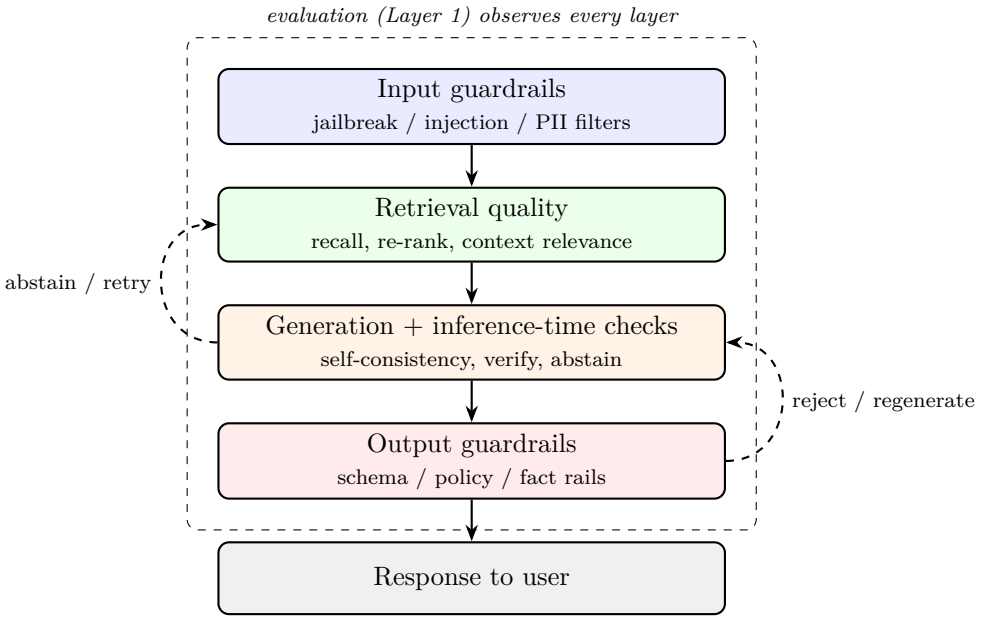

Figure 1: Defense-in-depth reference architecture. Solid arrows are the request path; dashed arrows are corrective loops (abstention, regeneration). Evaluation (Layer 1, dashed frame) is not on the request path but instruments every layer, producing the metrics that tune the others.

not measured). Their residual is $0.3^3 = 0.027$, which beats a single stronger filter at $\epsilon = 0.1$, but it costs roughly three inferences instead of one, the tradeoff the budget objective below makes explicit. Crucially, this compounding buys nothing for a class that only one of the three filters can even see: redundancy shrinks a term, it does not create coverage. Figure 2 plots this decay and contrasts it with the additive coverage effect.

Writing C for the set of failure classes, p_c for the base rate of class c, and $L(c)$ for the layers redundantly covering it, the overall residual is a sum over classes of a per-class product,

$$R \;\approx\; \sum_{c \in C} p_c \, \rho_c \prod_{i \in L(c)} \epsilon_{i,c}, \qquad \rho_c \geq 1.$$

Coverage decides which terms are nonzero; redundancy shrinks each surviving term. The factor ρ_c is where the linchpin assumption is made honest: $\rho_c = 1$ is the idealized independent case (the clean product), and $\rho_c > 1$ measures how much correlated failures inflate the residual when a hard input defeats several of class c's filters at once. Because ρ_c is a property of the inputs and the filters jointly, it must be *measured*, not assumed to be 1; carrying it explicitly is the standing argument for making redundant layers *mechanistically different* (a deterministic validator and an LLM judge fail on different inputs, pushing ρ_c toward 1).

Cost is spent where risk is. The cheap deterministic layers (input and output guardrails) run on every request; the expensive layers (multi-sample inference-time checks) run selectively, gated by a confidence or stakes signal. A system that runs Chain-of-Verification on every request has misallocated its compute; one that runs it only on flagged requests approaches the same reliability at a fraction of the cost.

This reframes reliability engineering as *budget allocation*. Weighting each class in R by the harm h_c of a failure in it, the goal is to choose the filter placement $L(\cdot)$ that minimizes expected residual

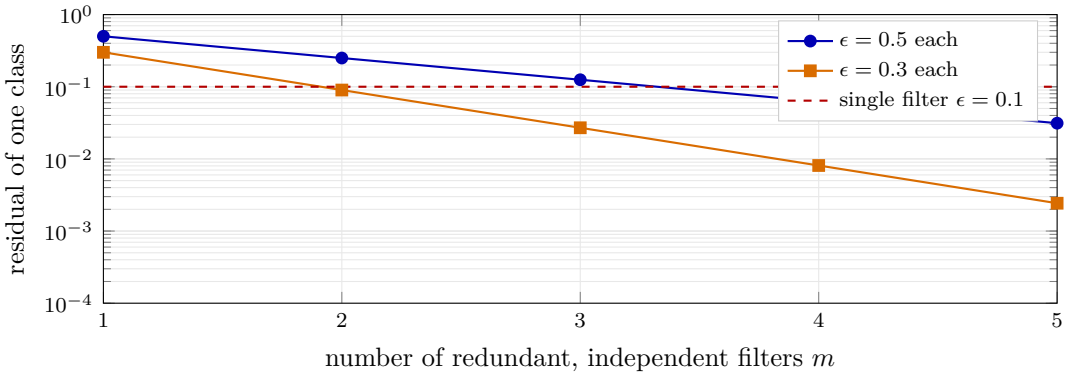

Figure 2: Redundancy within a single failure class is multiplicative: m independent filters of per-filter miss rate ϵ leave residual ϵ^m (log scale). Three $\epsilon = 0.3$ filters (orange) reach 0.027, below a single $\epsilon = 0.1$ filter (dashed), at the cost of running three filters. This decay applies only *within* a class the filters share; adding a filter for a *new* class instead removes that class's term from R additively (coverage). Rates are illustrative, not measured.

harm subject to a per-request cost ceiling,

$$\min_{L(\cdot)} \sum_{c \in C} p_c \, h_c \, \rho_c \prod_{i \in L(c)} \epsilon_{i,c} \quad \text{s.t.} \quad \sum_i \mathbb{1}[i \text{ runs}] \, \text{cost}(i) \le B,$$

where $\text{cost}(i)$ is filter i's latency-plus-token price and $\rho_c \ge 1$ is the same correlation factor as in R, kept inside the objective on purpose: an allocation optimized under the independence assumption $\rho_c = 1$ can be badly wrong precisely on the correlated, hard inputs that matter most, so the term must travel with the objective rather than be relegated to a caveat. Selective execution enters as the observation that a filter need not run on every request: gating an expensive filter on a cheap uncertainty signal lowers its amortized cost without changing its ϵ on the inputs that matter. We do not solve this program. We observe that the field optimizes individual $\epsilon_{i,c}$ (a better retriever, a better judge) without ever writing down the objective they feed into, nor collecting the p_c, $\epsilon_{i,c}$, h_c, and ρ_c needed to estimate it. That objective, and its data, is, to our knowledge, still largely absent from both the research literature and production practice.

A worked micro-instantiation. To show the objective is runnable rather than merely aspirational, consider a toy pipeline with three failure classes and two candidate filters (all numbers illustrative). Class rates and per-failure harm are: format violation ($p = 0.10$, $h = 1$), intrinsic hallucination ($p = 0.05$, $h = 5$), and jailbreak ($p = 0.02$, $h = 20$). Filter A is a cheap deterministic validator (cost $= 1$) that catches format violations well ($\epsilon = 0.05$) but is blind to the other two ($\epsilon = 1$). Filter B is an LLM judge (cost $= 4$) that partially catches hallucination and jailbreak ($\epsilon = 0.3$ on each) but not format. Under a per-request budget $B = 5$ we can afford both, and assuming near-independence ($\rho_c \approx 1$) the expected residual harm is

$$\underbrace{0.10 \cdot 1 \cdot 0.05}_{\text{format}} + \underbrace{0.05 \cdot 5 \cdot 0.3}_{\text{halluc.}} + \underbrace{0.02 \cdot 20 \cdot 0.3}_{\text{jailbreak}} = 0.005 + 0.075 + 0.120 = 0.20.$$

Had the same budget instead bought two copies of the judge B (dropping the validator), format violations would run unguarded ($0.10 \cdot 1 \cdot 1 = 0.10$) while the doubled judge drives the other two down by 0.3^2, for a total of $0.10 + 0.0225 + 0.036 = 0.16$ *only if* the two judge passes fail independently; at a realistic $\rho \approx 3$ for two copies of the *same* model that becomes $0.10 + 0.0675 + 0.108 = 0.28$,

worse than the mixed allocation. The lesson the arithmetic makes concrete: mechanistically diverse filters beat redundant identical ones once ρ is honest, and the cheapest deterministic layer is almost never the one to drop.

9 Discussion, Limitations, and Open Problems

Limitations, stated plainly. This is a synthesis and position paper, and we are careful not to claim more. The two load-bearing quantitative objects, the residual model R and the budget program of Section 8, are *illustrated* with pedagogical constants, not *validated* on a measured pipeline; we present no case study estimating real p_c, $\epsilon_{i,c}$, h_c, or the correlation factors ρ_c. The contribution is therefore a framework and a vocabulary for reasoning about composition, not evidence that the framework predicts a specific system's failure rate. We believe the honest framing is exactly this, a practitioner's map that the field can now instrument, and the single most valuable next step is one end-to-end case study on a real RAG or agentic pipeline that measures the quantities the model assumes, especially the cross-layer correlation ρ_c it currently only names.

The layered framing also clarifies what is solved and what is not.

Evaluator trust is the bottleneck. Every layer is tuned against a metric, and the metrics increasingly come from LLM judges [5, 26]. We lack a rigorous, cheap way to bound how much to trust an automated evaluator on a novel distribution. Until we do, reported reliability gains rest on a foundation that is itself unvalidated.

Calibrated abstention is underdeveloped. Models carry self-knowledge signal [12], but converting it into a well-calibrated, task-appropriate policy that abstains or defers to a human at the right moment remains ad hoc.

Agentic pipelines invert the arithmetic. Every mechanism in this paper is single-turn, yet the multiplicative composition that *helps* within a redundantly covered class turns hostile the moment a system chains steps, as it does in agents that interleave reasoning and acting over a trajectory [25]. If an agent takes n dependent actions and each succeeds independently with probability $1-\delta$, end-to-end success is $(1-\delta)^n$ and decays geometrically (Figure 3): a per-step error rate of $\delta = 0.05$, unremarkable in isolation, yields only a 0.60 success rate over a ten-step trajectory. The link back to the framework is direct: the per-step failure rate δ is precisely the single-turn residual of Section 8, $\delta = R = \sum_c p_c \rho_c \prod_{i \in L(c)} \epsilon_{i,c}$, evaluated at one step. Every layer multiplies its ϵ into δ and so sets the *base* of the exponential, but no single-turn layer can touch the *exponent* n. Controlling n (shorter plans, checkpointing, recovery) and decorrelating consecutive steps therefore become first-order reliability levers that the single-turn literature surveyed here barely addresses. Extending the failure-to-layer map and the budget objective of Section 8 to whole trajectories, where the base rates p_c themselves compound with depth, is the most consequential direction this framing leaves open.

Failures are correlated ($\rho_c > 1$). The multiplicative-decay argument is exact only when $\rho_c = 1$, and the real world violates this: a hard input can defeat several layers at once. Measuring ρ_c empirically, and designing layers to be maximally decorrelated so that ρ_c stays near 1, is an open and practically urgent problem, and the term that a naive component-by-component optimization silently sets to 1.

There is no end-to-end reliability budget. The field optimizes components (a better retriever, a better judge) without a shared framework for allocating a fixed error/latency/cost budget across the whole pipeline. The reference architecture above is a sketch of what such a framework would

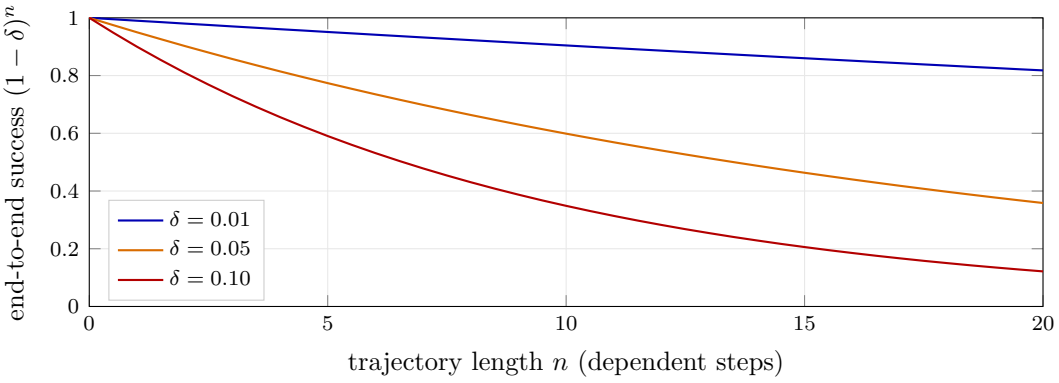

Figure 3: The trajectory inversion. When success requires n dependent steps to all hold, end-to-end success is $(1-\delta)^n$, where the per-step failure rate δ is the single-turn residual R of Section 8. The layers of this paper lower δ (shift between curves) but cannot change n (position along a curve); at $\delta = 0.05$, ten steps already cost 40% of successes.

instrument, not a solution.

10 Conclusion

Reliable LLM applications are not built by discovering one decisive technique; they are engineered by composing many imperfect ones. We organized the field's sprawling toolbox into four composable layers, evaluation, retrieval quality, inference-time reliability, and guardrails, mapped common failures to the layer that most cheaply catches them, and argued that reliability emerges from defense in depth: cheap filters, ordered by cost, where distinct filters extend coverage across failure classes and redundant, independent filters drive the residual of a shared class down multiplicatively. The framing turns a reading list into an architecture and exposes the discipline's real frontier, not any single missing method, but trustworthy evaluation, calibrated abstention, reliability that survives lengthening agent trajectories, and a principled way to spend a fixed reliability budget across the whole stack.

References

[1] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

[5] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. RAGAS: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[6] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

[7] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[9] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[10] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

[11] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[12] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

[13] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *International Conference on Learning Representations (ICLR)*, 2023.

[14] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023.

[16] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[17] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[18] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[19] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[20] Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, 2023.

[21] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[22] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024.

[23] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP*, 2021.

[24] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[25] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[26] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.