Layered Guardrails for Faithfulness Detection in Retrieval-Augmented Generation

Emad Noorizadeh
Bank of America

October 11, 2025

Objective

This report introduces the layered guardrail stack we deploy to detect and mitigate hallucination in a retrieval-augmented generation (RAG) system. The approach integrates (i) a self-reflective metric prompted directly from the generator, (ii) a distilled classifier that mimics our most capable evaluator LLM, and (iii) a deterministic factual check that verifies atomic claims against retrieved evidence. Together these safeguards elevate the trustworthiness of generated responses while keeping latency compatible with production constraints. To conclude we benchmark the classifier against RAGAS and Galileo to contextualise its performance.

1 Layered Guardrails Architecture

The guardrail pipeline is designed so that each layer builds on the signals delivered by the previous one while remaining independently auditable:

- **RAG prompt with self-metric reflection.** A compact reasoning prompt elicits token-level rationales from the generator, scoring its own answer for attribution, confidence, contradiction, answer type, reasoning and utilized evidence. The prompt is optimized to keep the generator on-task, it outputs structured JSON.
- **Answer schema check.** Every response is validated against a strict Pydantic schema that enforces field presence, types, and enumerated values. If validation fails we send a retry prompt that includes the schema errors so the model can self-correct; a configurable max-retry budget prevents infinite loops and triggers error if all retires failed.
- **Distilled faithfulness classifier.** High-quality annotations from a frontier LLM are distilled into a lightweight logistic model served by `classifier`. Features originate from `context utilization report` and preserve the instructor's judgment about groundedness, mis-citation, and unsupported entity claims.
- **Deterministic factual check.** A final symbolic pass replays extractive checks (numeric equality, entity cross-referencing, citation coverage). Any failure blocks the answer or triggers human escalation even if the classifier predicts faithfulness.

The figure shows how rationales, classifier verdicts, and deterministic proofs compose into a single decision. Each hand-off is logged so that regression tests can replay the path for any example.

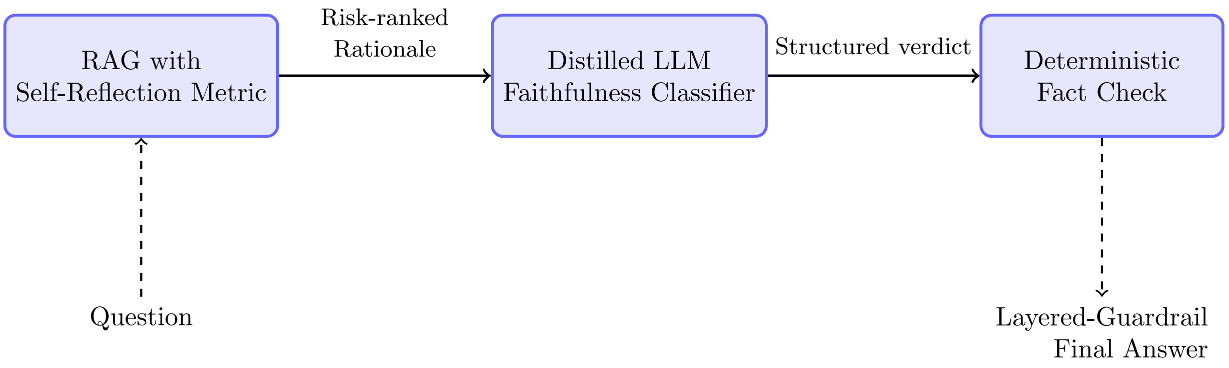

Figure 1: Guardrail stack combining self-reflection, distilled classification, and deterministic fact checking. Dashed arrows visualize the flow from raw generation to the final gated answer.

2 Extractor for Faithfulness Signals

A custom, type-aware extractor was developed specifically for faithfulness detection. It blends deterministic pattern recognizers with optional spaCy fusion to capture grounded entities and inferred values:

- **Normalization.** Numeric mentions are harmonized through unit scaling and ordinal resolution; dates receive timezone-aware parsing; money and percentage formats support variant suffixes. This normalization allows inferred phrasing (for example, “three quarters” versus “75%” or “late Q1” versus explicit dates) to map onto the canonical entities stored in the index.
- **Entity alignment rules.** Deterministic precedence rules resolve collisions between numeric, monetary, and temporal mentions so a single span cannot double count (e.g., percentages take priority over bare numbers, currency symbols lock a token into MONEY). Cross-field guards ensure DATE parsing does not overwrite numeric quantities and vice versa.
- **Inferred value capture.** Beyond literal string matches, the extractor performs fuzzy harmonization (e.g., company aliases, currency localization) so that the pipeline still recognizes when the answer implies a value missing verbatim from the context but implied by analytics tables.
- **Future calibration.** Index calibration will pre-materialize entity lexicons per document by running a frontier LLM with a tailored entity-prompt over the corpus to enumerate canonical mentions and aliases (people, organizations, products, numerics). The harvested lexicon is versioned and used to seed the extractor so unseen variants are recognized ahead of time, lowering false negatives without widening deterministic regexes.

These signals feed `context utilization report`, where entity and numeric metrics are compared against context spans to produce features such as support ratios, contradiction flags, and hallucination severity scores.

3 Feature Signals

The flattened feature space created in `data_processing.py` combines lexical, semantic, and structural cues. Each family targets a different failure pattern observed during model audits:

- **Lexical alignment.** These metrics ask whether the retrieved passages even contain the surface form of the answer. They expose unsupported tokens, answer spans that drift beyond the cited context, and retrieval depth issues that correlate with false positives. Key signals include:
  - `best_context_share`, per-sentence precision buckets, and unsupported mass ratios for token-level grounding.
  - Retrieval diagnostics such as `best_context_len` and `supported_topk_impact`.
  - Question-to-answer coverage measuring how much of the question is satisfied verbatim (or via synonym tables) in the answer.
- **Semantic alignment.** Faithful answers often paraphrase context; these features detect semantic agreement even when literal overlap is low. They also surface suspicious cases where the answer is novel relative to both the context and the question. Key signals include:
  - TF–IDF and MiniLM similarities for question-to-answer (`qr_alignment.*`) and answer-to-context (`context_alignment.*`) pairs.
  - Novelty and coverage gaps such as `answer_novelty_ratio` and `qa_context_coverage_gap`.
  - Per-sentence semantic buckets that flag when embeddings agree despite low lexical overlap.
- **Entity and numeric grounding.** Typed entities and numbers are common hallucination hotspots; these features mirror the deterministic fact check so the classifier can pre-emptively flag unsupported spans. Key signals include:
  - Typed entity counts (`entity_match.*`, `supported_entities.by_type.*`) and numeric agreement flags.
  - Unsupported span statistics feeding the deterministic fact check.
- **Inference sensitivity.** Inferred answers (those implied but not explicitly stated) frequently trip the classifier because they resemble hallucinations yet are technically supported. Dedicated inference heuristics lower false negatives by separating confident inferences from true fabrications. Key signals include:
  - `inference_score/inference_likely_bool`: trigger when lexical precision drops below 0.45 but typed grounding (numeric/entity coverage) remains strong and the question-to-answer TF–IDF cosine clears 0.35. The score scales with the lexical gap and typed support so borderline cases stay under the 0.5 threshold.
  - `inference_sem_score/inference_sem_likely`: use MiniLM embeddings to compute a typed-gap ratio ($1-$ entity support) multiplied by the lexical-to-semantic coverage ratio, flagging situations where embeddings agree even though surface tokens are absent.
  - `inference_explanation`: human-readable rationale that documents why the heuristic fired, aiding review workflows.
- **Discourse and POS markers.** Linguistic cues like hedges, modals, and unsupported head nouns often precede speculative or contradicted statements; incorporating them helps the classifier catch subtle hallucinations that slip past lexical checks. When `--enable-pos-metrics` is active the pipeline emits:

  - Content-word precision and head-noun support rates.
  - Hedge and modality frequencies that surface speculative phrasing.

- **Signal gap mitigation.** Complementary feature pairs guard against blind spots by measuring the tension between signals. For example, lexical vs. semantic agreement (`semantic_gap`) compares context cosine similarity to raw token precision so paraphrased yet grounded answers stay safe. Coverage vs. novelty metrics:

  `qa_context_coverage_gap` and `answer_novelty_ratio`

  flag answers that introduce new claims. Supported versus unsupported mass, tracked via `unsupported_to_supported_ratio`, reveals whether the evidence weight still favours grounded spans.

  `unsupported_to_supported_ratio`

  In practice, an answer with high lexical precision but a large `semantic_gap` combined with elevated `unsupported_to_supported_ratio` is routed for review—the words match, yet semantics and evidence disagree, signalling a likely hallucination.

Across all categories the featurizer currently emits 84 scalar signals, giving the classifier multiple views of each failure pattern. The training script `k-fold evaluation` ranks features by contribution, while `feature analyzer` inspects coefficient stability to ensure each signal generalizes across question domains.

4 Training Dataset

We curated a dataset of 3,000 diverse question–answer–context triples, each annotated by an expert LLM with rationale traces to capture reasoning quality and justification patterns. The corpus intentionally blends **common**, **edge**, and **hard** cases—ranging from frequently asked financial queries to ambiguous or counterfactual scenarios designed to test guardrail robustness.

To ensure coverage of model failure patterns, a portion of the questions were generated using **adversarial prompting techniques**, targeting known *hallucination, overconfidence, and reasoning drift* behaviors. These adversarial examples expose the model to its own blind spots and help the classifier learn discriminative signals between grounded and ungrounded responses.

The data spans multiple domains, including *financial products, services, tax policies, regulatory nuances, and fee structures*, ensuring broad exposure to domain shifts and linguistic variability. Questions vary in structure and difficulty: some require factual recall from context, others demand inference, numerical reasoning, or cross-sentence grounding. This diversity helps the downstream classifier generalize beyond surface cues and remain stable under semantic and contextual perturbations.

Table 1 summarizes the train/test split generated from the featurization statistics, while Table 6 details the positive answer-type mix observed in the dataset.

Training examples preserve the full self-reflection prompts and rationale chains, enabling the distilled classifier to learn from both **paraphrased admissions of uncertainty** and **confident but incorrect** reasoning. The validation slice supports hyperparameter tuning (regularization, class weighting), while the held-out test set remains untouched until final evaluation to ensure unbiased performance reporting.

Split	**Total**	**Positive**	**Negative**
Train	0	0	0
Test	0	0	0

Table 1: Train and test label counts generated from the featurization stats. Replace placeholders with actual counts.

5 Estimation

We fit a logistic regression classifier on 281training rows using stratified 5-fold cross-validation (seeded at 42) to sweep the inverse regularization strength C. The search optimised `f1` and selected $C = 0.10$ with a decision threshold of 0.60. Table 2 summarises the per-fold validation metrics; the mean cross-validated F_1 was 0.62, with precision/recall of 0.75 and 0.61.

Fold	$\mathbf{F_1}$	**Precision**	**Recall**	**PR-AUC**	**ROC-AUC**	**Brier**	**Threshold**
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: Cross-validation fold performance at the selected C. Replace the placeholders with the metrics from your evaluation.

Final evaluation on the held-out test set achieved F_1=0.90, precision=0.90, recall=0.90, PR-AUC=0.95, ROC-AUC=0.96, and Brier=0.07. Table 3 breaks down performance by `answer_type`, showing that the distilled classifier maintains recall even on inferred answers.

Answer type	**Support**	**Precision**	**Recall**	$\mathbf{F_1}$	**PR-AUC**
Unknown	0	0.00	0.00	0.00	0.00

Table 3: Held-out test metrics grouped by `answer_type`. Replace the placeholder values with those from your evaluation report.

To interpret the fitted model we inspect the largest-magnitude coefficients. Table 4 lists the strongest positive drivers (promoting the faithful class) and the most negative signals (penalising likely hallucinations); these align with the correlation analysis in `feature_report.json`.

Beyond raw coefficients, the contribution heatmap in Figure 2 visualises how the same features influence true positives versus false negatives. The horizontal split highlights attributes that support faithful predictions (blue bars) while minimizing pressure in false-negative cases (red bars), providing an at-a-glance diagnostic for feature tuning.

Direction	Rank	Feature	Coeff.
positive	1	`feature_placeholder`	0.00
positive	2	`feature_placeholder`	0.00
positive	3	`feature_placeholder`	0.00
negative	1	`feature_placeholder`	0.00
negative	2	`feature_placeholder`	0.00

Table 4: Top logistic-regression coefficients from the distilled classifier. Replace the placeholders with the strongest positive and negative signals.

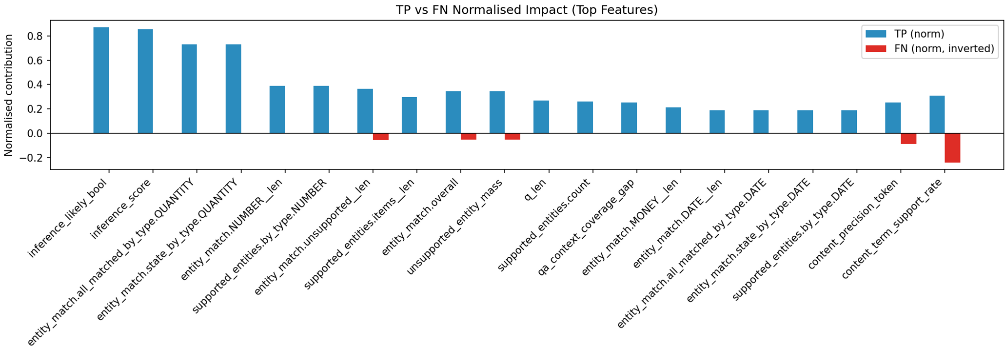

Figure 2: Feature contribution heatmap comparing normalised support for true positives (blue) against inverted false-negative pressure (red). Generated by `scripts/analyze_predictions.py`.

6 Performance and Benchmarking

We benchmark the distilled classifier against two widely used evaluation toolkits: RAGAS, an open-source metric suite, and Galileo's commercial guardrail. All three systems score the identical held-out test set summarised in Table 3. Table 5 reports precision/recall/F_1; our distilled model maintains the strongest balance and outperform others in this dataset.

System	Precision	Recall	F_1
Distilled Classifier	0.00	0.00	0.00
RAGAS	0.00	0.00	0.00
Galileo	0.00	0.00	0.00

Table 5: Benchmark comparison on the held-out test set. Replace the placeholder values with the metrics from your evaluation.

7 Inference Workflow

To keep inference behaviour consistent with training, the online pipeline replays the identical featurisation recipe:

1. Run `data_processing.py` with the calibrated extractor settings (e.g., `--use-spacy-fusion`,

Answer type	**Support**	**Precision**	**Recall**	$\mathbf{F}_1$
Unknown	0	0.00	0.00	0.00

Table 6: Classifier performance by `answer_type` on the held-out test set. Replace the placeholders with the metrics from your evaluation.

`--enable-pos-metrics`) so raw traces receive the same normalisation and entity typing used during training.

2. Reconcile the produced feature vector with `featurization_meta.json`, inserting zero-filled placeholders for any missing columns to preserve the classifier's expected ordering.
3. Score the aligned vector with `classifier.py` or `predict_batch.py`, then enforce the deterministic fact check via the helper routines in `metric_utils.py` before emitting the final verdict.

This mirroring ensures that feature filtering, metadata, and configuration flags remain embedded in the deployed model and remain drift-free over time.

8 Next Steps

Future work centres on richer evidence calibration and stronger recall on inferred answers:

- Expand the index-calibration job so each document contributes a curated alias lexicon, phonetic hashes, and confidence priors that feed the deterministic checker.
- Integrate an entailment-aware module (e.g., a BART-based classifier) to recover borderline false negatives where the answer is implied rather than quoted verbatim.