When Knowledge Changes: Metamorphic Testing of RAG Systems with Mutations

Jinhan Kim✉
Università della Svizzera italiana
Lugano, Switzerland
jinhan.kim@usi.ch

Samuele Pasini
Università della Svizzera italiana
Lugano, Switzerland
samuele.pasini@usi.ch

Paolo Tonella
Università della Svizzera italiana
Lugano, Switzerland
paolo.tonella@usi.ch

Abstract

Retrieval-Augmented Generation (RAG)-based LLM systems rely on external document corpora that can evolve and change over time. However, current evaluation methodologies (e.g., RAGAS) assess correctness against static snapshots, failing to detect faults when routine updates, factual changes, or noise alter the underlying data. We introduce a metamorphic testing framework that evaluates the consistency of RAG systems under corpus evolution. We formalise a fault taxonomy and 11 mutation operators that systematically perturb the system at both the pre-chunk (retrieval index) and post-chunk (retrieved context) levels. An empirical evaluation across five datasets and over 28k mutants reveals metamorphic violation rates of 4.9–10.2%. In a meta-evaluation against ground truth, our metamorphic oracle achieves F1 scores of 0.927–1.000, while the best RAGAS metric reaches only 0.570. Finally, we provide actionable insights into mitigating these faults through retrieval re-configuration, generator upgrades, and LLM-based reranking.

CCS Concepts

• **Software and its engineering**;

Keywords

Retrieval-Augmented Generation, Metamorphic Testing

ACM Reference Format:
Jinhan Kim, Samuele Pasini, and Paolo Tonella. 2026. When Knowledge Changes: Metamorphic Testing of RAG Systems with Mutations. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26), October 12–16, 2026, Munich, Germany*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3832783.3837543

1 Introduction

Retrieval-Augmented Generation (RAG) systems have become a dominant architectural pattern for deploying large language models (LLMs) in knowledge-intensive applications [7]. By retrieving documents from an external corpus and conditioning generation on the retrieved content, RAG systems aim to overcome the limitations of LLMs' parametric knowledge and provide answers grounded in external sources. This architecture is widely adopted in settings such as enterprise knowledge bases, compliance assistants, and proprietary documentation systems, where correctness depends on access to evolving document collections [10].

ASE '26, Munich, Germany

ACM ISBN 979-8-4007-2882-2/2026/10
https://doi.org/10.1145/3832783.3837543

Existing evaluation methodologies for RAG systems primarily assess snapshot performance [4, 12]. Benchmarks typically measure answer accuracy, retrieval quality, or grounding with respect to a fixed corpus representation. These evaluations are effective for comparing model configurations under controlled conditions, but they implicitly assume that the document corpus and retrieval pipeline remain unchanged. In practice, this assumption does not always hold: document corpora evolve continuously, facts are updated, rules change, evidence is added or removed, and documents are restructured, re-chunked, or re-indexed as part of system maintenance and evolution.

From a software engineering perspective, it is critical that a RAG system remains sensitive to the *state* of the corpora. The system should not only provide correct answers for a static snapshot but also demonstrate appropriate behavioural shifts (or stability) as the corpus undergoes semantically significant (or insignificant) changes. For example, when an irrelevant document is re-formatted, the system's output should remain unchanged; when a factual value is updated, the output should reflect the update; when supporting evidence is removed, the system should express uncertainty (e.g., abstain) rather than hallucinate. Whether current RAG systems satisfy such expectations remains unexplored.

To overcome this problem, we introduce a metamorphic testing paradigm specifically tailored for RAG systems. Rather than relying on a fixed ground-truth answer, we evaluate correctness by observing how system outputs shift in response to controlled semantic transformations of the corpus. We formalise these expectations as a set of Metamorphic Relations (MRs). These relations define the invariant properties of the system under corpus evolution, such as maintaining behavioural consistency when encountering irrelevant noise, updating answers appropriately when factual premises change, and explicitly acknowledging conflicts when contradictory evidence is injected.

Systematically driving this evaluation requires formalising how RAG systems fail. Based on a fault taxonomy that we derived by synthesising failure modes from prior empirical studies of RAG robustness [8, 14, 18–20], we design 11 mutation operators applied across two operational scopes: *pre-chunk* mutations that perturb the upstream document index, and *post-chunk* mutations that alter the downstream retrieved context directly. We then execute the RAG pipeline across the original and mutated context, checking the outputs against our defined MRs. Any violation serves as an indicator of semantic inconsistency, pointing to a potential weakness of the RAG system.

We evaluate our framework on five datasets, executing over 28k mutants across both scopes. Our study yields three principal findings. First, the framework detects faults across all five datasets, with violation rates ranging from 4.9% to 10.2%. The two mutation

scopes expose complementary faults (Jaccard overlap 12–45%), confirming the diagnostic value of testing at both the index and context levels. Second, in a meta-evaluation against ground-truth correctness, our MR oracle achieves F1 scores of 0.927–1.000, whereas the best-performing RAGAS metric [4] reaches only 0.570 F1. Third, repair effectiveness varies substantially across datasets. The union of retrieval-budget increases, a generator upgrade, and LLM-based reranking resolves 43.1% of evaluated faults. Union repair rates range from 47.3% to 63.0% across the four financial datasets but reach only 10.2% on RepLiQA, pointing to failure modes that neither broader retrieval, stronger generation, nor LLM-based reranking can fully address.

In summary, this paper makes the following contributions:

- We introduce a metamorphic testing framework for RAG systems under corpus evolution, driven by 11 mutation operators across two scopes.
- We conduct an empirical study across five datasets demonstrating that our MRs expose silent failures that RAGAS metrics overlook, validated by a human study and oracle stability analysis.
- We analyse the repairability of detected faults, showing that LLM-based reranking complements retrieval and generator interventions and that their union repairs 43.1% of detected faults.

2 Background

This section summarises established concepts and practices relevant to this work.

2.1 Retrieval-Augmented Generation (RAG)

RAG refers to a class of systems that combine a language model with an external document collection [5]. Given a user query, a retrieval component selects a set of documents from the corpus, and a generation component (e.g., LLM) produces an output conditioned on both the query and the retrieved documents. RAG systems are commonly used when the information required to answer queries is not fully captured in the model's parameters, or when access to external, domain-specific, or up-to-date knowledge is required. The document corpus is therefore treated as an integral part of the system rather than as static training data.

The standard RAG procedure consists of two primary phases: document indexing and output generation. During the offline indexing phase, the corpus is prepared by parsing and segmenting documents into smaller textual units called *chunks*. These chunks are transformed into high-dimensional vectors via an embedding model [3] and stored in a vector database. This spatial arrangement is fundamental to the system's behaviour, as proximity within the vector space indicates semantic similarity between pieces of information. In the subsequent online inference phase, the system processes a user query by embedding it with the same model to identify and retrieve the *top-k* most similar chunks using a distance measure like cosine similarity. These chunks serve as the essential context for the system's decision-making and are integrated into a structured prompt. The process concludes when the LLM generates a response that is specifically conditioned on both the original query and this retrieved evidence.

2.2 Evaluation of RAG Systems

Frameworks like RAGAS [4] and ARES [12] propose automated, reference-free evaluation metrics that approximate human judgements using LLM-as-a-judge. For example, at the retrieval level, *Context Precision* calculates the fraction of retrieved chunks that are genuinely relevant to the query. Similarly, *Context Relevance* evaluates the context at the sentence level by measuring the ratio of sentences strictly necessary to answer the query against the total number of sentences retrieved. At the generation level, *Response Relevancy* estimates how well the generated answer addresses the original query. It achieves this by prompting an auxiliary LLM to generate new questions based on the generated answer and calculating their semantic similarity to the original query. Finally, *Faithfulness* evaluates whether the answer is actively supported by the retrieved context. It decomposes the response into atomic statements and calculates the proportion of these statements backed by the retrieved chunks, which is crucial for identifying hallucinated claims.

While these metrics are useful for comparing systems under controlled conditions, they provide limited insight into testing how a RAG system behaves as its underlying knowledge base evolves. This highlights the need for evaluation frameworks that assess semantic consistency upon knowledge evolution, ensuring the system remains robust under realistic corpus perturbations.

2.3 Metamorphic Testing

Metamorphic Testing (MT) is a testing methodology developed for systems where test oracles are unavailable, incomplete, or impractical [13]. Instead of checking individual outputs for correctness, MT specifies relations that should hold between outputs produced under related inputs or environments. Violations of Metamorphic Relations (MRs) indicate unexpected or inconsistent system behaviour, even when individual outputs may appear plausible.

In this work, MT provides a basis for evaluating RAG systems by comparing system behaviour across different corpus states without assuming a fixed oracle. Instead of checking whether an output is absolutely correct against a static ground truth, we define a set of MRs that specify how the system's outputs should relate when the document corpus undergoes controlled semantic changes. This relational approach allows us to define correctness as the satisfaction of specific predicates across original and mutated corpora. By checking these relationships, we expose structural faults and reasoning errors that may remain undetected by standard evaluation methods.

3 System Model and Fault Taxonomy

In this section, we first formalise the RAG architecture as a composite pipeline of retrieval and generation functions governed by a set of system parameters (Section 3.1). We then introduce a fault taxonomy that classifies robustness failures based on the system's inability to maintain semantic consistency under controlled perturbations to its corpus or configuration (Section 3.2).

3.1 System Model

The overview of our approach is depicted in Figure 1. We treat the RAG pipeline as a stateful process where the transition from

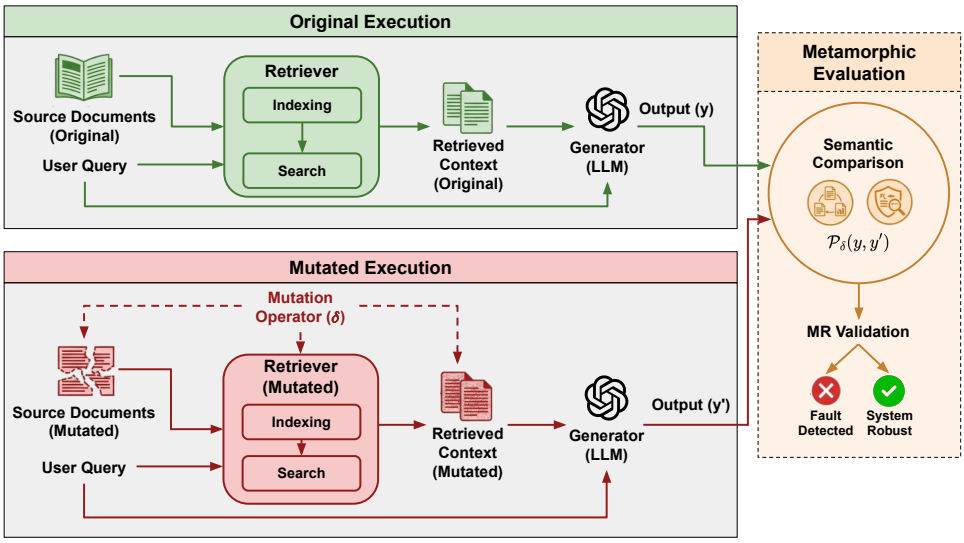


Figure 1: Overview of our metamorphic testing framework for RAG systems. The original execution (top) establishes a baseline by processing source documents and a user query through standard indexing and search to produce an original answer (y). The mutated execution (bottom) introduces a mutation operator (δ) that perturbs the system. The metamorphic evaluation component performs a semantic comparison between two outputs (y and y'). A violation of the metamorphic relation ($\neg\mathcal{P}_\delta$) indicates a detected fault, whereas semantic consistency suggests system robustness.

input query to output answer is mediated by the retrieval of evidence from the corpus. By introducing systematic perturbations to the source documents, retrieved context, or system settings, we would (or would not) observe the resulting shifts in the output. This differential execution allows us to isolate the causal chain of a failure: a fault is not merely a wrong answer, but a deviation from the expected semantic relationship between two executions.

We formalise a RAG system $\mathcal{S}$ as a composite function of a retriever $\mathcal{R}$ and a generator $\mathcal{G}$, operating over a dynamic corpus C. To capture the highly configurable nature of these pipelines, we denote the overall system configuration as $\Theta = \langle \theta_{\mathcal{R}}, \theta_{\mathcal{G}} \rangle$, where $\theta_{\mathcal{R}}$ represents retrieval hyperparameters (e.g., top-k, chunk size, or embedding model) and $\theta_{\mathcal{G}}$ represents generation hyperparameters (e.g., LLM temperature, LLM model version).[1] Given a query q, the system produces an output $y = \mathcal{S}_\Theta(q, C) = \mathcal{G}_{\theta_{\mathcal{G}}}\left(q, \mathcal{R}_{\theta_{\mathcal{R}}}(q, C)\right)$

In real-world deployments, neither C nor Θ is static. The corpus evolves through updates, segmentation changes, or noise accumulation, while system configurations are frequently tuned to optimise performance. This environmental and configurational volatility renders traditional test oracles (fixed $\langle q, a \rangle$ pairs) insufficient, as the correct answer a may change or become invalid under different system states.

To address this oracle problem, we adopt MT. We define correctness not as an absolute match against a static ground truth, but as the satisfaction of MR. Let δ be a mutation operator that perturbs the system state such that $\langle C', \Theta' \rangle = \delta(C, \Theta)$. A robust RAG system must satisfy a specific metamorphic relation $\mathcal{P}_\delta$ between the original output y and the post-mutation output y':

$$\forall q, C, \Theta: \quad \mathcal{P}_\delta\big(\mathcal{S}_\Theta(q, C), \mathcal{S}_{\Theta'}(q, C')\big)$$

In this framework, a *fault* is defined as a violation of this predicate ($\neg\mathcal{P}_\delta$). For example, if δ injects irrelevant noise into the corpus, $\mathcal{P}_\delta$ requires semantic invariance ($y \approx y'$). Conversely, if δ updates a fact in the corpus or severely restricts $\theta_{\mathcal{R}}$, causing essential chunks to drop, $\mathcal{P}_\delta$ dictates a corresponding output update or a fallback response. The following taxonomy classifies these violations based on the semantic nature of the failure.

3.2 Fault Taxonomy

To systematically characterise failures in RAG systems, we define a fault model based on the system's inability to maintain semantic consistency under perturbations. Our taxonomy was constructed by synthesising failure modes reported in prior empirical studies of RAG robustness. We began from the RAG noise taxonomy of Wu et al. [18], which identifies noise types that degrade LLM performance, and extended it with findings from studies on irrelevant context sensitivity [14], knowledge conflicts between retrieved and parametric knowledge [19], positional bias in long contexts [8], and retrieval configuration sensitivity [20]. We consolidated these into three fault classes: (1) *Noise Sensitivity Faults*, extending the RAG noise taxonomy established by Wu et al. [18]; (2) *Format Intolerance Faults*; and (3) *Architectural Brittleness Faults*, a classification addressing structural brittleness in retrieval pipelines. This taxonomy serves as the foundation for the mutation operators defined in Section 4.

[1] We distinguish the generator parameters ($\theta_{\mathcal{G}}$) from the retrieval parameters ($\theta_{\mathcal{R}}$) because $\theta_{\mathcal{G}}$ could theoretically be mutated to expose faults. However, our proposed mutation operators do not alter $\theta_{\mathcal{G}}$ because this work primarily targets the retrieval component of the RAG pipeline. Nevertheless, one of our research questions (RQ3) explores adjusting $\theta_{\mathcal{G}}$ to *fix* a fault once it has been localised to the generator rather than the retriever (see Section 6.3).

3.2.1 Noise Sensitivity Faults. This category encompasses faults where the generator (i.e., LLM) fails to correctly filter or arbitrate external information, leading to output corruption.

- *Context Overfitting Fault*: A logic fault where the system treats topically related but information-poor padding as valid evidence. In production environments, this manifests when retrieval systems surface content optimised for search engines or advertisements that share keywords with the query but lack factual substance. Shi et al. [14] demonstrate that LLMs are highly susceptible to such irrelevant context, often prioritising retrieved distractors over relevant knowledge.
- *Knowledge Conflict Fault:* A state inconsistency fault where the system fails to resolve contradictory facts. This reflects the common real-world challenge of *stale data*, where a knowledge base contains both outdated legacy documents and recent updates (e.g., v1 vs. v2 manuals). Wu et al. [19] demonstrate that systems frequently exhibit 'context bias', overriding correct internal priors to align with contradictory retrieved evidence. A robust system must exhibit explicit conflict awareness, signalling or resolving the discrepancy.
- *Input Robustness Fault*: A failure to handle minor character-level perturbations, resulting in significant degradation of retrieval precision or generation quality [18]. This fault is critical in pipelines processing OCR-scanned legacy documents, noisy speech-to-text transcripts, or user-generated content containing informal spelling.

3.2.2 Format Intolerance Faults. Although recent studies suggested that mixing data types should theoretically help models distinguish between plain text and core data, many RAG systems still fail under these conditions [18]. Such faults typically occur because the system lacks the flexible parsing logic required to process unexpected structural changes.

- *Heterogeneous Data Fault*: The system can fail or hallucinate when processing mixed-modality content. This is a pervasive issue in technical domains where documentation frequently interleaves natural language with code snippets (Python, SQL), JSON configurations, or tabular financial data.
- *Syntactic Filtering Fault*: The system attempts to reason over unintelligible, garbled token sequences instead of discarding them as noise. This simulates the faults in data cleaning pipelines, where HTML artefacts, encoding errors (mojibake), or truncated data packets pollute the retrieval index, leading to garbage-in-garbage-out behaviour.

3.2.3 Architectural Brittleness Faults. This category describes faults where the system's correctness is tightly coupled to specific implementation details or parameters of the retrieval pipeline ($\theta_{\mathcal{R}}$), rather than the semantic content of the knowledge base. A robust RAG system should exhibit *operational invariance*, producing consistent outputs regardless of choices in data discretisation or retrieval configuration.[2]

- *Segmentation Boundary Fault:* A fault where the system's output is unstable with respect to the specific discretisation (chunking) of the source text. In practice, fixed-size windowing often arbitrarily severs semantic dependencies, splitting a disclaimer from its clause or a subject from its verb. This sensitivity is empirically linked to the "lost-in-the-middle" phenomenon identified by Liu et al. [8], where models fail to retrieve relevant information solely because the chunking strategy shifted the evidence away from the edges of the context window.
- *Configuration Overfitting Fault:* This represents a dependency fault where correctness is strictly tied to specific chunk alignments within the index or a fixed retrieval budget defined in $\theta_{\mathcal{R}}$. It demonstrates that the system is brittle and overfitted to a narrow range of configurations. Xu et al. [20] highlight that architectural choices in retrieval depth can unexpectedly degrade performance, exposing this underlying lack of generalisation.

[2]However, note that we scope this expectation to non-disruptive changes; radical configuration changes that fundamentally alter the retrieved evidence would justifiably produce different outputs (see Section 4.5 for details).

4 Mutation Operators and Test Oracles

We define a set of 11 mutation operators to realise the fault taxonomy defined in Section 3.2. In the following sub-sections, after defining the scope of mutation, we introduce our novel mutation operators, followed by a description of the metamorphic relations used as oracles.

4.1 Mutation Scope

We evaluate RAG robustness across two distinct operational stages, pre-chunking and post-chunking. By perturbing the system at different stages, we can assess how changes to the knowledge base and the retrieved context independently influence the reliability of the generated output.

4.1.1 Pre-chunk Scope (System-Level Mutation). This scope performs system-level mutation by modifying the raw documents or retrieval configurations *upstream* of the ingestion pipeline. Formally, we define the mutated system state as $\langle C', \theta'_{\mathcal{R}} \rangle = \delta_{pre}(C, \theta_{\mathcal{R}})$, resulting in a perturbed output $y' = \mathcal{G}_{\theta_{\mathcal{G}}}(q, \mathcal{R}_{\theta'_{\mathcal{R}}}(q, C'))$. This perturbation alters the documents in the underlying corpus or the retrieval parameters (such as embedding models or chunking strategies), requiring a full re-indexing of the knowledge base. Unlike simple input fuzzing for LLMs, this modifies the index structure and knowledge representation, simulating the operational reality of continuous document edits, policy updates, or architectural migrations.

4.1.2 Post-chunk Scope (Input-Level Mutation). This scope performs input-level mutation (analogous to standard input fuzzing) targeting the retrieved context passed from the retriever to the generator. While less realistic, it can provide a more efficient alternative to pre-chunk mutation as it bypasses the re-indexing of the corpus or the re-computation of embeddings. Formally, we define the perturbed output as $y' = \mathcal{G}_{\theta_{\mathcal{G}}}\left(q, \delta_{post}(\mathcal{R}_{\theta_{\mathcal{R}}}(q, C))\right)$, where δ_{post} perturbs the retrieved context before it reaches the generator. Mutations are injected *downstream* of the retrieval step, modifying only the transient prompt while keeping the retrieval index C and configurations $\theta_{\mathcal{R}}$ and $\theta_{\mathcal{G}}$ pristine. This approach serves as a controlled experiment: by guaranteeing the mutation is present in the prompt, we isolate the generator and stress-test its reasoning capabilities independently of any retrieval variance. Consequently,

this scope simplifies fault localisation; if a failure occurs under post-chunk mutations, the behavioural change can be attributed to the generator.

4.2 Noise Mutation Operators

These operators inject textual perturbations to expose *Noise Sensitivity* and *Format Intolerance* faults.

Semantic Noise Operator (SeN). SeN targets *Context Overfitting Faults* by injecting off-topic sentences from a static pool of unrelated facts. Pre-chunk injection scatters distractors throughout the document, shifting chunk boundaries and polluting the index to test if the retriever maintains precision against irrelevant signals. Post-chunk injection appends off-topic text to retrieved chunks, testing the LLM's ability to filter irrelevant trailing context.

Datatype Noise Operator (DN). DN targets *Heterogeneous Data Faults* by injecting foreign fragments (code, URLs, JSON) into prose. Fragments are drawn from a predefined pool of code snippets and inserted at sentence boundaries. Pre-chunk injection embeds these into the corpus structure, testing vector space robustness against mixed modalities. Post-chunk injection injects fragments into retrieved chunks individually, validating if the LLM can parse mixed-type inputs without hallucination.

Illegal Sentence Noise Operator (ISN). ISN targets *Syntactic Filtering Faults* by inserting syntactically malformed 'garbage' sentences constructed by randomly sampling and concatenating words from a predefined vocabulary of common English tokens (e.g., articles, verbs, nouns), producing syntactically incoherent text. Pre-chunk injection distributes this noise across the document, potentially causing the retriever to index unintelligible chunks. Post-chunk injection appends garbled text to retrieved chunks, testing the LLM's exception handling and garbage-in-garbage-out resilience.

Counterfactual Noise Operator (CN). CN targets *Knowledge Conflict Faults* by injecting statements that contradict ground truth. To produce realistic contradictions, the operator prompts a separate LLM to generate a plausible counterfactual statement given the original fact with the user query. In pre-chunk injection, contradictions are embedded in the corpus; critically, the retriever may return the contradiction without the original fact, simulating stale data scenarios. In post-chunk injection, a contradiction is generated for and appended to the retrieved chunks, forcing the LLM to arbitrate immediate logical conflicts between parametric priors and external evidence.

Supportive Noise Operator (SuN). SuN targets *Context Overfitting Faults* by adding topically related but information-void fillers. The operator heuristically extracts key terms (e.g., domain nouns) from the source text and inserts them into predefined sentence templates, producing filler that sounds on-topic but contains no specific facts. Pre-chunk injection dilutes the corpus, challenging dense retrievers that may rank filler-heavy chunks highly due to semantic overlap. Post-chunk injection appends fillers to retrieved chunks, testing if the LLM is distracted by relevant-sounding padding.

Orthographic Noise Operator (ON). ON targets *Input Robustness Faults* by applying character-level typos. Pre-chunk injection degrades the index, testing fuzzy matching between clean queries and misspelled documents. Post-chunk injection preserves retrieval but forces the LLM to reason over degraded surface forms.

4.3 Chunking Mutation Operators

These operators alter the segmentation strategy to expose *Segmentation Boundary Faults*.

Chunk Merge (CM) & Chunk Split (CSp) Operators. CM aggregates adjacent chunks into fewer, larger index entries, altering the granularity of the retrieval index. CM is only applicable in pre-chunk mode; at the post-chunk stage, the retriever has already returned individual chunks, so there are no adjacent chunks to merge. Conversely, CSp subdivides chunks. Pre-chunk injection fragments the index, increasing the risk of relevant content falling below retrieval thresholds; post-chunk injection forces the LLM to synthesise information across a highly fragmented context list.

Chunk Shift Operator (CSh). CSh re-chunks the document starting from an offset (e.g., 50%). Pre-chunk injection performs global re-chunking, creating entirely new boundary alignments across the corpus. Post-chunk injection performs a local token shift, trimming tokens from the start of a retrieved chunk and prepending them to the next. This operator directly provokes the "lost-in-the-middle" phenomenon [8] by arbitrarily shifting the position of key evidence within the context window.

4.4 RAG Configuration Operators

These operators perturb the execution environment to evaluate the system against *Configuration Overfitting Faults*. By directly modifying the retrieval hyperparameters $\theta_{\mathcal{R}}$ rather than the corpus text, these operators are exclusively pre-chunk: they require re-indexing or re-retrieval, which has no analogue at the post-chunk stage where the retrieved context is already fixed.

Embedding Model Swap (EM). EM replaces the retrieval hyperparameter in $\theta_{\mathcal{R}}$ (e.g., swapping `text-embedding-3-small` for `large`) to evaluate whether system performance is coupled to a specific vector space. A robust RAG implementation should exhibit operational invariance across different embedding geometries.

Top-k Perturbation (TK). TK modifies the retrieval hyperparameter in $\theta_{\mathcal{R}}$ by altering the number of chunks passed to the generator. Restricting k evaluates whether the generator can maintain precision when relying on fewer retrieved chunks, whereas expanding k tests whether it can isolate relevant evidence from redundant or distracting context. However, beyond its role as a mutation operator, we note that changing k may also *repair* a fault. As a repair operator, increasing k may recover missing evidence, while decreasing k may filter out distractions. The same type of top-k change can therefore induce or repair a fault depending on its context, consistent with the inversive relationship between bugs and patches observed by Kim et al. [6]. This motivates our evaluation of retrieval-budget changes as a repair intervention in RQ3 (see Section 6.3).

4.5 Metamorphic Relationships as Test Oracles

We define MRs that verify the logical consistency between RAG system executions. Specifically, we implement the metamorphic predicate $\mathcal{P}_\delta$ (defined in Section 3.1) to enforce principles of *semantic invariance* or *controlled variance* across mutations. Our oracles

are categorised into two classes based on the expected transformation of the system's response:

- *Invariance Relations (Semantic-Preserving):* For a mutation operator δ_p that preserves the semantic essence of the corpus to answer the query, a robust system must exhibit operational invariance. We expect $y \approx y'$, where $\approx$ denotes semantic equivalence.
- *Variance Relations (Semantic-Altering):* For a mutation operator δ_a, e.g., that injects counterfactual facts related to the query, the system must update its output to reflect the new state of the knowledge base. We expect $y' \neq y$, specifically where y' aligns with the injected fact.

It is important to note that, in reality, the classification of an MR as semantic-preserving or semantic-altering is a function of both the mutation operator type and its specific parameters. For example, with the ISN operator, a 5% noise injection ratio is intended to be semantic-preserving, whereas a 95% injection (replacing most content with garbled text) is likely semantic-altering. Evaluating both the mutation's effect and the system's response requires judgements that go beyond simple string matching. We employ an LLM-as-a-Judge [21] (detailed in Section 5.2.3) in two roles: (1) a *semantic preservation judge* that verifies, for each mutant, whether the factual evidence required to answer the query remains intact after the perturbation, determining the actual semantics-preserving or -altering status rather than relying on the operator's intended category; and (2) an *answer equivalence judge* that assesses whether y and y' are semantically equivalent given the query and context, implementing the $\approx$ / $\neq$ relations above.

When the RAG system *abstains*, the verdict is assigned deterministically without invoking the LLM judge. For semantics-preserving mutations, abstention constitutes an MR violation; for semantics-altering mutations, abstention is treated as MR satisfaction.

5 Experimental Setup

5.1 Research Questions

We structure our empirical study around the following three research questions:

RQ1 (Fault Detection): *Can our metamorphic framework detect RAG systems' faults?* We quantify the system's susceptibility to noise and structural perturbations by investigating the rates of MR violations per mutation operator. Additionally, we conduct a comparative analysis between pre-chunk (system-level) and post-chunk (input-level) mutations to assess their respective effectiveness and identify the overlap in the faults detected.

Lastly, we conduct a human validation study to evaluate the reliability of our automated oracle. By measuring the alignment between human annotators and the LLM-as-a-Judge regarding both the semantic preservation of the mutation and the equivalence of the generated answers, we aim to prove that our automated evaluation framework is trustworthy.

RQ2 (Comparison): *Do our MRs reveal the faults that existing RAG evaluation metrics overlook?* We compare our MRs with four representative snapshot metrics from RAGAS. This comparison does not treat RAGAS as a robustness oracle: RAGAS evaluates an individual answer and its retrieved context, whereas our MRs evaluate the relationship between two executions before and after a controlled mutation. We therefore examine whether snapshot metrics detect the same evolution-induced faults exposed by relational testing. Using ground-truth correctness as the meta-evaluation oracle, we compute precision, recall, and F1-score for each evaluator and analyse their fault overlap using Venn diagrams. We also assess the stability of the MR oracle on RepLiQA, where answer-equivalence judgements require LLM-based evaluation. We stratify-sample 100 cases by mutation scope, operator, initial verdict, and semantic-preservation label, then repeat the final MR judge verdict five times per case.

RQ3 (Repair): *Can identified faults be mitigated by retrieval, generation, or reranking interventions?* We investigate the persistence of identified faults under three repair interventions: increasing the retrieval budget, upgrading the generator model, and inserting an LLM-based reranking stage before generation. For pre-chunk faults, we evaluate direct retrieval with $k = 3$ where applicable and with $k = 5$, together with the reranking. For post-chunk faults, we evaluate an upgrade from GPT-5-mini to GPT-5.4 and reranking over the mutated retrieved context. The reranking strategy partitions its candidates into at most 30 evidence units, uses an LLM to score each question-unit pair from 0 to 5, and selects up to six of the highest-ranked units for answer regeneration.

5.2 Experimental Procedure

Our evaluation employs a multi-stage pipeline designed to isolate the causal impact of mutations on RAG behaviour. We utilise datasets (detailed in Section 5.4) comprising a query q, a corpus C (documents or relevant information), and a GT answer a_{gt}. While a primary advantage of our metamorphic approach is its independence from GT labels in production, we leverage a_{gt} in the study to facilitate a rigorous meta-evaluation of *evaluators*.

5.2.1 Filtering for Context Dependency. To ensure our test suite targets the retrieval pipeline rather than the generator's parametric memory, we apply a *closed-book filter*. We first execute the generator in isolation (without C): $y_{closed} = \mathcal{G}(q)$. We retain only those triplets $\langle q, C, a_{gt} \rangle$ in the dataset where the closed-book response y_{closed} is *incorrect*, i.e., $y_{closed} \neq a_{gt}$, while with C, i.e., generating the answer using the full RAG pipeline, $y_{base} = \mathcal{S}(q, C)$ successfully produces the ground-truth answer a_{gt}, i.e., $y_{base} \approx a_{gt}$. This step ensures that the retrieved evidence is a *necessary condition* for correctness, allowing us to attribute subsequent failures to the perturbation δ.

5.2.2 Metamorphic Execution. For each filtered instance, we treat the successful baseline output y_{base} as our reference point (y in Figure 1). We then apply a mutation δ to generate a post-mutation output y'. Unlike traditional benchmarks that compare y' to a static ground truth a_{gt}, our metamorphic oracle evaluates the logical consistency between y_{base} and y'. We say a fault is detected by our framework if the answer pair $\langle y_{base}, y' \rangle$ violates the metamorphic predicate $\mathcal{P}_\delta$.

5.2.3 Use of LLM-as-a-Judge. The complexity of evaluating answer pairs (e.g., comparing a_{gt} with y_{base}, or y_{base} with y') depends heavily on the output format. For numerical outputs in our financial dataset, T2-RAGBench (see Section 5.4), we perform direct comparison to determine correctness. However, when the generator

produces natural language outputs, assessing logical equivalence becomes non-trivial.

To address this, we employ an LLM-as-a-Judge [21]. The judge is provided with the query q, the retrieved contexts, information about the perturbation δ, and the respective answer pairs to determine their relationship under $\mathcal{P}_\delta$. In addition to evaluating the system's output, we utilise the LLM-as-a-Judge to verify the nature of the mutation itself. Specifically, the judge determines whether a perturbation is semantic-preserving (where the factual evidence required to answer q remains intact) or semantic-altering. This automated verification is critical because the disruptiveness of the mutation operators is highly sensitive to their parameterisation. For instance, operators such as SeN can vary from subtle stylistic changes to highly disruptive perturbations if a high volume of noise is injected into the document.

Although the final verdict of the judge is binary, we utilise Likert scoring-based prompting to capture logical and linguistic nuances, a method shown to be more effective than simple prompting [17]. For example, in the case of semantic-altering mutations, the judge evaluates how effectively the system updates its response to reflect new or counterfactual information on a 1–5 scale. High scores of 4–5 indicate successful adaptation, where y' clearly reflects the mutated context or appropriately acknowledges conflicting information, satisfying the $\mathcal{P}_\delta$ relation. Conversely, scores of 1–2 indicate a failure to adapt, where y' remains essentially identical to y_{base} despite the change in evidence, resulting in an MR violation.

We acknowledge that relying on the LLM-as-a-Judge introduces a potential threat to validity due to hallucinations of the judges.To assess this threat, we conduct two complementary analyses. First, two authors independently annotated a random sample of 100 RepLiQA mutation pairs, for both semantic preservation and answer equivalence. Second, we assessed oracle stability using 100 RepLiQA cases stratified by mutation scope, operator, initial verdict, and semantic-preservation label, repeating the final MR judgement five times for each case. Section 6.1 reports the human agreement and oracle stability results.

5.3 Meta-Evaluation for Enabling Comparison

To quantify the diagnostic accuracy of our framework, we introduce a meta-evaluation procedure that assesses the effectiveness of our framework against established RAG evaluation metrics, response relevancy, faithfulness, context precision, and context recall, provided by the RAGAS framework [4]. This stage effectively serves to evaluate the evaluators by measuring how reliably each oracle detects system failures. While a core advantage of MR is its ability to operate without a ground-truth oracle, we selectively re-introduce a_{gt} here as an objective baseline to determine absolute factual correctness under mutation. With GT, we map the verdicts of both RAGAS and our MR framework into a formal classification of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN), and corresponding recall, precision, and F1-score.

For our framework, we compare the metamorphic verdict ($\mathcal{P}_\delta$) against the logical relationship between the post-mutation output (y') and the ground truth (a_{gt}). For the RAGAS metrics, we define a fault detection as any score falling below a set threshold. This mapping allows us to identify *silent faults*. For instance, if a semantic-preserving mutation results in an output y' that contradicts a_{gt} while a RAGAS metric maintains a high score, the metric suffers an FN, failing to register a structural fragility that our MR framework is designed to expose.

5.4 Datasets

We use two primary benchmarks to evaluate and compare our metamorphic framework for testing RAG systems across different domains and reasoning requirements.

5.4.1 T2-RAGBench. The financial question-answering scenario is based on T2-RAGBench [16], a benchmark designed to evaluate RAG over large collections of financial documents containing both textual and tabular information.

T2-RAGBench includes four datasets: ConvFinQA [1], FinQA [1], TAT-DQA [23], and VQAonBD [11] that cover heterogeneous financial sources such as annual reports, regulatory filings, and financial statements. All four datasets provide predefined test queries and verified answers. Test queries in these datasets typically require multi-step reasoning over retrieved evidence, such as locating specific tables or passages, extracting numerical values, and performing comparisons or arithmetic operations.

5.4.2 RepLiQA. We utilise the RepLiQA benchmark [9], which is a question-answering dataset specifically curated to benchmark LLMs on unseen reference content. It comprises *synthetic* reference documents crafted by human annotators, depicting imaginary scenarios, people, and places across 17 distinct categories. Because these fictional entities and events are entirely absent from the internet, they could not have been included in the pre-training corpora of existing LLMs. Each sample in RepLiQA includes a reference document, a specific question about the document's topic, and a ground-truth answer derived directly from the text. By relying on purely synthetic content, RepLiQA guarantees that accurate answers can only be generated if the RAG system successfully retrieves the correct document and the LLM accurately comprehends the provided context, rather than relying on memorised facts.

5.4.3 Filtering, Sampling, and Cost. Although datasets in our evaluation were designed to preclude data contamination during LLM training, we nevertheless subject all datasets to the filtering procedure described in Section 5.2.1. We retain only the context dependent triplets $\langle q, C, a_{gt} \rangle$. This filtering reduces the initial datasets to a pool of qualified instances: 1,868 for ConvFinQA, 3,139 for FinQA, 3,412 for TAT-DQA, 7,791 for VQAonBD, and 6,888 for RepLiQA.

However, even with the refined pools, evaluating the entire population remains economically (or computationally) infeasible to us. Each base instance yields 11 mutated executions. Applying these across two mutation modes does not strictly double the count: since three operators are inapplicable to post-chunk mode (Section 4), we execute $11 + 8 = 19$ total mutants per triplet. Every mutant requires LLM API calls for both execution and evaluation. Our framework uses 2–3 LLM calls per mutation test: one for RAG generation on the mutated corpus, one for the semantic preservation judge, and one for the answer equivalence judge (the latter is replaced by exact numeric comparison on financial datasets, reducing the count to 2). For running RAGAS, each of its four metrics invokes the LLM

Table 1: Pre-chunk and Post-chunk MR Violation Rates (%) with Fault Overlap (Jaccard %)

	RepLiQA			ConvFinQA			FinQA			TAT-DQA			VQAonBD		
MO	**Pre**	**Post**	**J**	**Pre**	**Post**	**J**	**Pre**	**Post**	**J**	**Pre**	**Post**	**J**	**Pre**	**Post**	**J**
CM	8.1	–	–	5.0	–	–	2.3	–	–	4.8	–	–	5.1	–	–
CSh	9.0	8.3	41	8.0	6.0	11	8.3	6.0	10	7.3	5.3	6	11.0	1.3	6
CSp	5.0	9.0	27	9.3	4.3	17	7.7	2.0	4	5.0	2.3	10	14.7	2.0	6
CN	11.5	18.0	5	19.7	23.3	12	18.0	26.0	15	24.7	22.3	11	27.6	27.0	17
DN	7.4	9.0	69	4.1	4.3	25	4.0	2.7	6	4.7	3.3	35	1.3	2.3	25
EM	2.3	–	–	13.6	–	–	4.0	–	–	0.0	–	–	12.5	–	–
ISN	8.2	9.3	72	6.3	4.3	12	7.6	2.7	19	3.4	3.3	12	1.9	2.0	22
ON	9.0	9.7	87	3.7	4.3	50	3.3	2.4	13	3.7	3.3	31	2.3	1.9	38
SeN	11.4	9.3	62	8.4	4.3	21	6.0	2.7	10	4.3	3.0	25	2.9	1.0	22
SuN	7.8	9.3	52	5.3	4.3	35	3.6	3.0	20	3.5	3.0	29	1.4	2.0	25
TK	9.3	–	–	2.1	–	–	2.8	–	–	4.9	–	–	3.8	–	–
Avg.	**8.1**	**10.2**	**45**	**7.8**	**6.9**	**18**	**6.2**	**5.9**	**12**	**6.0**	**5.8**	**15**	**7.7**	**4.9**	**15**

Table 2: Mutation parameters used in the experiments.

MO	Pre-chunk	Post-chunk
SeN	$0.20n_s$ sentences	1 sentence/chunk
DN	$n_s/3$ fragments	$n_s/3$ fragments
ISN	$0.15n_s$ sentences	1 sentence/chunk
CN	1 passage/document	1 passage/chunk
SuN	$0.20n_s$ sentences	1 sentence/chunk
ON	Error rate 0.01	Error rate 0.01
CM	Merge factor 2	N/A
CSp	Split factor 2	Split factor 2
CSh	Offset 0.5 (128 words)	10-token shift
EM	`small` → `large`	N/A
TK	$k = 1 \rightarrow 3$	N/A

n_s denotes the number of sentences. Sentence-based quantities are rounded down with a minimum of one. EM changes between the `text-embedding-3-small` and `text-embedding-3-large` models.

independently (*Faithfulness*: 2 calls for statement extraction and inference; *Answer Relevancy*: 3 calls for question generation; *Context Precision* and *Context Recall*: 1 call each), totalling 7 LLM calls per evaluation point. For a campaign of N entries and M operators, we use $N{\times}M{\times}3$ LLM calls, while RAGAS uses $N{\times}M{\times}7$. We note that this cost breakdown is not a comparative efficiency claim between ours and RAGAS, which serve different evaluation purposes; rather, it quantifies the cumulative API cost that necessitated sampling instead of exhaustive evaluation. To keep the experiment tractable while preserving statistical validity, we restrict the scope of the evaluation. From each of the five filtered datasets, we sample $n = 300$ base instances. This yields $300 \times 19 = 5{,}700$ mutants per dataset, and a total of 28,500 mutants across five datasets. All results are reported based on these samples and mutants.

5.5 Configurations

We run closed-book filtering ten times, retaining only those triplets that consistently satisfy these conditions across all trials. In the study, the default top-k is set to 1. Table 2 reports the parameter settings used in the experiment. For pre-chunk mutations, *SeN* and *SuN* insert sentences corresponding to 20% of the original sentence count, *ISN* uses 15%, and *DN* inserts one fragment per three sentences. At post-chunk scope, *SeN*, *ISN*, and *SuN* insert one sentence per retrieved chunk, while *DN* inserts one fragment for every three sentences. *CN* adds one counterfactual passage per document or retrieved chunk, and *ON* applies a character-level error probability of 0.01. The structural operators use a merge or split factor of 2, while *CSh* uses a 0.5 offset before chunking and a 10-token shift after retrieval. Finally, *EM* changes the embedding model from `text-embedding-3-small` to `text-embedding-3-large`, and *TK* increases k from 1 to 3. In the repair study, we do not evaluate $k = 3$ as a repair for faults induced by *TK* as it reproduces the mutated retrieval setting; we still evaluate these faults using $k = 5$ and LLM-based reranking. Sentence-derived quantities are rounded down with a minimum of one.

Default RAG generation, counterfactual generation, closed-book filtering, semantic-preservation judging, and answer-equivalence judging use OpenAI's GPT-5-mini[3]. As the GPT-5 family is composed of reasoning models, the API does not support user configurable sampling parameters such as temperature or top-p; we therefore use the API defaults. We use `text-embedding-3-small`[4] as the embedding model. We use a RAGAS fault threshold of 0.3. While the artefact also includes results for thresholds of 0.0 and 0.5, these variations do not significantly alter the findings. All experiments are conducted on a machine equipped with an Apple M3 Max and 36 GB of RAM, running Tahoe 26.2.

6 Results

6.1 RQ1 (Fault Detection)

Table 1 reports MR violation rates and fault overlap (Jaccard index, J) by mutation operator and scope. Average violation rates range from 6.0% to 8.1% for pre-chunk mutations, and from 4.9% to 10.2% for post-chunk mutations. *CN* consistently produces the highest violation rates across all datasets and both scopes, reaching 27.6% (pre) and 27.0% (post) on VQAonBD, showing that injecting counterfactual content directly degrades both retrieval relevance and generation faithfulness. Chunking operators *CSh* and *CSp* also produce

[3] `gpt-5-mini-2025-08-07`
[4] https://platform.openai.com/docs/models/text-embedding-3-small

elevated pre-chunk rates (e.g., 14.7% for *CSp* on VQAonBD), because altering chunk boundaries before indexing changes which content is retrieved. In contrast, surface-level operators (*ON*, *SuN*, *DN*) yield lower violation rates overall, indicating that RAG pipelines tolerate minor formatting and typographic perturbations well.

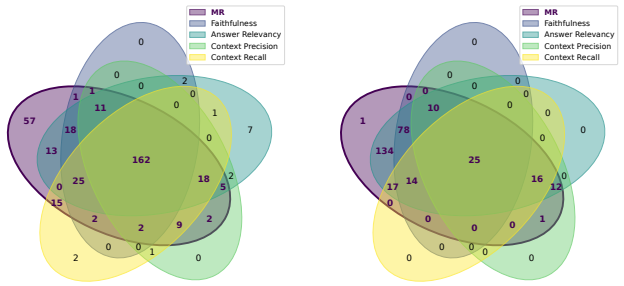
To illustrate, consider two RepLiQA faults (i.e., True Positives). When asked "What was a significant obstacle Emily faced during her marathon training?", the baseline system correctly answered "a minor ankle sprain." After a semantics-preserving *CSh* mutation that merely shifted chunk boundaries, the system instead answered "balancing a full-time job with her running schedule," retrieving a different passage fragment and producing a factually different answer despite no change to the underlying content. Conversely, when a semantics-altering *CN* mutation injected counterfactual information about a yoga studio's new offering, the system was asked "What new offering did ZenSpace Yoga launch in early 2024?" and still answered "virtual reality meditation classes" verbatim, ignoring the contradictory evidence now present in the retrieved context.

The Jaccard column (J) reveals that pre-chunk and post-chunk mutations largely expose different faults. Average overlap is only 12–18% on four of five datasets, with RepLiQA being the exception at 45%. Surface-level operators (*ON*, *ISN*, *DN*) show relatively high overlap because these perturbations are too minor to meaningfully alter retrieval. Both scopes end up probing the same generator-side sensitivity, and the fault sets converge. In contrast, *CN* exhibits very low overlap. Overall, the low average overlap confirms that the two mutation scopes are complementary: using both would broaden fault coverage beyond what either scope achieves alone. RepLiQA's higher overlap (45%) can be explained by its design, where each query targets one short document, so the retriever is likely to return similar chunks regardless of whether the mutation is applied before or after chunking.

To validate the automated verdicts, two authors independently labelled 100 randomly-sampled RepLiQA mutation pairs for answer equivalence and semantic preservation. Inter-annotator agreement was 98% ($\kappa = 0.92$) and 99% ($\kappa = 0.96$), respectively.[5] Each annotator agreed with the LLM judge 93–95% of the time ($\kappa = 0.68$–0.77). Most disagreements (7 out of 8) occurred when the LLM judged answers as equivalent but annotators deemed them different, typically involving partial matches where the mutated answer omitted details or added precision (e.g., "late 2023" vs. "September 12, 2023"). Annotator confidence on these disputed cases averaged 2.50 out of 5, compared to 4.58 on agreed items, indicating that disagreements cluster at genuinely ambiguous boundaries rather than reflecting systematic bias.

To assess whether stochastic variation in judging affects the detected violations, we repeated the final MR verdict five times for 100 stratified RepLiQA cases. 94 cases received unanimous verdicts, and the majority verdict agreed with the initial verdict in 97 cases. SATISFIED/VIOLATED flips occurred in six cases, indicating that judgement instability is limited but not absent.

[5] κ denotes Cohen's kappa, a chance-corrected measure of inter-rater agreement [2].



(a) RepLiQA **(b) ConvFinQA**

Figure 2: Fault-overlap Venn diagrams for RepLiQA and ConvFinQA.

Answer to RQ1: Our metamorphic framework detected faults across all five datasets (violation rates 4.9–10.2%), with pre- and post-chunk mutations exposing distinct faults (Jaccard overlap 12–45%), confirming both scopes are needed. A further human study validated the automated verdicts. Human validation yielded 93%–95% agreement with the automated judge. Across repeated oracle calls, 94% of cases received unanimous verdicts and 97% retained the initial verdict under majority voting.

6.2 RQ2 (Comparison)

Table 3 compares our MR-based framework against four RAGAS metrics in a meta-evaluation against ground-truth correctness. MR achieves near-perfect F1 across all datasets: 0.927 on RepLiQA and 1.000 on others.[6] In comparison, the best-performing RAGAS metric is *Context Recall*, which reaches its highest F1 of 0.570 on RepLiQA. All other RAGAS metrics remain below 0.46 F1 across all datasets.

Figure 2 illustrates the fault-overlap structure for RepLiQA and ConvFinQA. The remaining datasets exhibit similar patterns and are provided in our artefact (see Section 9). The diagrams depict only true-positive cases. Although metrics such as *Answer Relevancy* appear to offer broad coverage, Table 3 shows that all RAGAS metrics suffer from low precision and recall, so their apparent TP-only coverage overstates actual reliability. RAGAS metrics also largely overlap with each other, while MRs capture additional faults that RAGAS misses entirely.

These results demonstrate that metamorphic testing provides a more reliable diagnostic signal than score-based evaluation. By checking behavioural consistency under controlled perturbations rather than estimating quality from a single response, MRs achieve both high recall and high precision.

Answer to RQ2: Our metamorphic framework substantially outperforms RAGAS metrics, achieving F1 scores of 0.927–1.000 compared to 0.083–0.570 for RAGAS. The fault-overlap analysis shows that RAGAS metrics are largely redundant with each other, whereas MRs additionally detect faults that RAGAS misses.

[6] Note that the perfect F1 on the financial datasets is expected: their answers are single numeric values, so MR validation via exact match coincides with meta-evaluation with GT, leaving no room for classification error (see Section 8).

Table 3: Meta-evaluation: MR vs. RAGAS metrics. Recall, Precision, and F1 computed from TP/TN/FP/FN classifications against ground-truth correctness.

	RepLiQA			ConvFinQA			FinQA			TAT-DQA			VQAonBD		
Metric	**Recall**	**Prec.**	**F1**	**Recall**	**Prec.**	**F1**	**Recall**	**Prec.**	**F1**	**Recall**	**Prec.**	**F1**	**Recall**	**Prec.**	**F1**
Faithfulness	0.575	0.379	0.456	0.373	0.077	0.128	0.625	0.085	0.150	0.605	0.103	0.176	0.492	0.086	0.147
Answer Relevancy	0.707	0.220	0.336	0.992	0.070	0.131	1.000	0.060	0.113	0.990	0.060	0.114	0.987	0.065	0.122
Context Precision	0.554	0.369	0.443	0.181	0.065	0.096	0.401	0.103	0.164	0.113	0.066	0.083	0.155	0.089	0.113
Context Recall	0.641	0.514	0.570	0.209	0.060	0.093	0.645	0.099	0.171	0.254	0.090	0.133	0.272	0.118	0.165
Ours	0.894	0.963	**0.927**	1.000	1.000	**1.000**	1.000	1.000	**1.000**	1.000	1.000	**1.000**	1.000	1.000	**1.000**

6.3 RQ3 (Repair)

Table 4 reports all repair configurations. For pre-chunk faults, direct retrieval with $k = 3$ and $k = 5$ repairs 32.5% and 32.5%, respectively, while reranking repairs 30.8%. For post-chunk faults, upgrading the generator repairs 32.6%, while reranking repairs 23.2%. Altogether, the considered configurations repair 43.1% of faults. These aggregate results mask substantial variation across datasets. Union repair rates range from 47.3% to 63.0% across the four financial datasets but reach only 10.2% on RepLiQA. For pre-chunk faults, direct retrieval achieves 28.8% to 51.8% on the financial datasets, while both retrieval budgets repair 4.0% on RepLiQA. Reranking similarly achieves 30.2% to 49.1% on the financial datasets but only 6.5% on RepLiQA. For post-chunk faults, the generator upgrade repairs 37.0% to 48.2% on the financial datasets but only 7.8% on RepLiQA, while reranking achieves 26.1% to 35.5% and 5.8%, respectively. The consistently low rates on RepLiQA indicate faults that broader retrieval, stronger generation, and reranking cannot adequately address.

Repairability also varies by mutation operator. Among pre-chunk faults, *EM* is the most repairable through direct retrieval with $k = 5$ (49.1%), while *CSp* benefits most from reranking (49.6%). In contrast, *CN* remains difficult for both interventions, with repair rates of 16.4% and 19.7%, respectively. For post-chunk faults, *CN* benefits most from the generator upgrade (37.3%) but least from reranking (16.6%), whereas *SuN* achieves the highest reranking repair rate (36.9%). These differences show that the considered interventions address distinct failure modes rather than providing a uniformly effective repair.

As an example of a successful repair, a semantics-preserving *SeN* post-chunk mutation caused the system to abstain ("I cannot answer this question based on the provided context"); upgrading the generator recovered the correct answer with full detail. Conversely, a semantics-altering CN mutation that injected a counterfactual about an app lacking currency conversion features persisted after repair: the stronger model still adopted the contradictory evidence, producing an answer that directly opposed the ground truth.

While the achieved repair rates remain partial, we note two limitations of this analysis. First, we evaluate only three general-purpose repair interventions, whereas developers may devise additional strategies based on their system and domain knowledge. Second, we do not know what proportion of the detected faults practitioners would consider worth repairing in a real deployment. Despite these limitations, the 43.1% union repair rate indicates that the evaluated interventions can address a substantial subset of the detected faults.

Answer to RQ3: Repair is useful but partial. For pre-chunk faults, setting $k = 3$, $k = 5$, and reranking repair 32.5%, 32.5%, and 30.8%, respectively. For post-chunk faults, the generator upgrade and reranking repair 32.6% and 23.2%. The union repairs 43.1% of evaluated faults, but the 10.2% repair rate on RepLiQA shows that some corpus-evolution faults persist across all interventions.

7 Related Work

Frameworks such as RAGAS [4] and ARES [12] evaluate retrieval and generation quality for individual query–answer pairs against a *fixed* corpus snapshot. While effective for point-in-time benchmarking, they evaluate outputs in isolation and cannot capture whether a system behaves consistently as the corpus evolves. Our work introduces a relational, corpus-evolution-aware paradigm that surfaces failures invisible to these static metrics.

MetaRAG [15] is closer to our setting as it also applies metamorphic testing to RAG systems, but it places the testing boundary at the response level: it mutates factoids extracted from a generated answer and checks whether the variants are supported by the retrieved context. In contrast, our system under test is the RAG pipeline under corpus evolution. Pre-chunk mutations alter documents, chunking, the index, or retrieval configuration before re-executing ingestion, retrieval, and generation, while post-chunk mutations isolate generator behaviour after retrieval. Thus, MetaRAG checks whether an existing answer is supported by its evidence, whereas our framework tests whether a RAG pipeline behaves consistently when the knowledge state changes.

Prior work shows that LLMs are easily distracted by irrelevant retrieved passages [14], exhibit positional bias when key evidence appears mid-context [8], and suppress correct parametric knowledge when retrieved content contradicts it [19]. The most directly related work is Wu et al. [18], which taxonomises RAG noise types empirically. Our fault taxonomy (Section 3.2) extends theirs by adding Format Intolerance and Architectural Brittleness classes, and makes each class an executable mutation operator with an automatic oracle, rather than a descriptive analysis.

A separate line of work investigates adversarial attacks on retrieval based systems. Zhong et al. [22] show that injecting a small number of adversarial passages into a dense retrieval corpus can mislead retrieval models with high success rates, even generalising to unseen queries and domains. PoisonedRAG [24] extends this

Table 4: Fault repair success by dataset and scope. Each cell reports repaired/applicable faults (rate); the union counts a fault as repaired if any applicable configuration succeeds.

Dataset	Pre-chunk			Post-chunk		All
	$k = 3$	$k = 5$	Rerank	GPT-5.4	Rerank	Union
RepLiQA	10/248 (4.0%)	11/276 (4.0%)	18/276 (6.5%)	19/243 (7.8%)	14/243 (5.8%)	53/519 (10.2%)
ConvFinQA	102/212 (48.1%)	113/218 (51.8%)	107/218 (49.1%)	80/166 (48.2%)	59/166 (35.5%)	242/384 (63.0%)
FinQA	67/196 (34.2%)	59/205 (28.8%)	62/205 (30.2%)	61/142 (43.0%)	37/142 (26.1%)	164/347 (47.3%)
TAT-DQA	78/190 (41.1%)	78/204 (38.2%)	71/204 (34.8%)	51/138 (37.0%)	43/138 (31.2%)	182/342 (53.2%)
VQAonBD	86/208 (41.3%)	102/215 (47.4%)	86/215 (40.0%)	52/118 (44.1%)	34/118 (28.8%)	189/333 (56.8%)
Overall	343/1,054 (32.5%)	363/1,118 (32.5%)	344/1,118 (30.8%)	263/807 (32.6%)	187/807 (23.2%)	830/1,925 (43.1%)

to end-to-end RAG, achieving 90% attack success by injecting as few as five crafted texts per target question into a knowledge base of millions. These works consider adversarial robustness, whereas our framework tests robustness under corpus evolution (reformatting, updates, noise). The two perspectives are complementary: poisoning attacks expose security vulnerabilities, while our metamorphic approach surfaces engineering faults that arise during corpus maintenance.

8 Threats to Validity

Internal validity. The LLM-based judge may misclassify semantic preservation or answer equivalence. Our human study (Section 6.1) found agreement between 93%–95%, although we acknowledge that the random sample did not cover all case categories evenly: semantics-altering cases comprised 12.0% of the sample compared with 12.3% of the evaluated RepLiQA population, while violated cases comprised 4.0% compared with 9.1%. We do not control or measure all sources of stochasticity across the complete pipeline: repeating mutation generation, e.g., the generation of counterfactuals, may produce a different execution. Our stability analysis isolates one component: we held each sampled mutation, retrieved context, and generated answer fixed and repeated only the final MR judgement five times. We observed unanimous verdicts in 94% of cases, while the majority verdict agreed with the initial verdict in 97%. The four financial datasets use exact numeric comparison for answer equivalence, reducing their reliance on the judge. The closed-book filter may retain queries answerable by chance; we mitigate this by running it ten times.

External validity. Our evaluation spans five datasets in two domains but uses a single RAG architecture and LLM. Different retrieval strategies or models may yield different fault profiles. We sample 300 base instances per dataset due to the cost of running both our framework and RAGAS across all mutation operators and scopes (28k mutants total). However, the consistent patterns observed across five independently sampled datasets provide confidence in the robustness of the findings. In future work, the cost of mutation campaigns could be reduced by prioritising the most effective operators (e.g., *CN*, *CSh*) based on the violation rates reported in our results.

Construct validity. The RQ2 meta-evaluation uses ground-truth to classify faults. For RepLiQA this again relies on the LLM judge, but exact-match results on financial datasets corroborate the same findings. A potential concern is circularity in the meta-evaluation: the filtering procedure (Section 5.2) already verifies that the baseline answer y matches the ground truth a_{gt}, so for semantics-preserving mutations the MR check ($y' \approx y$) and the GT check ($y' \approx a_{gt}$) are correlated by transitivity. We note three mitigating factors. First, the filtering is necessary for experimental validity, not to advantage MR: without it, mutations would be applied to already-incorrect baselines, making fault detection uninterpretable. Both MR and RAGAS are evaluated on the same filtered instances. Second, the correlation does not hold for semantics-altering mutations, where the GT check verifies whether y' reflects the injected counterfactual, independently of $y \approx a_{gt}$. Third, RepLiQA empirically demonstrates non-trivial evaluation: MR achieves 0.927 F1, not 1.000, confirming that the LLM-based equivalence judgement ($y' \approx y$) and the GT comparison ($y' \approx a_{gt}$) can genuinely disagree when answers are natural language rather than single numeric values.

9 Conclusion

We presented a metamorphic testing framework for evaluating RAG systems under corpus evolution. Rather than assessing snapshot correctness, the framework defines metamorphic relations that verify behavioural consistency across 11 mutation operators applied at two scopes: pre-chunk (index-level, exercising the full pipeline) and post-chunk (context-level, isolating the generator). Across five datasets, our framework detects faults (violation rates from 4.9% to 10.2%) that standard RAGAS metrics miss, with human validation confirming the automated verdicts. The two mutation scopes expose complementary faults. Retrieval-budget increase, generator upgrade, and LLM-based reranking together resolve 43.1% of detected faults. While this is encouraging, their limited effectiveness on RepLiQA points to future work on alternative repair strategies.

Data Availability

The implementation and the data are available.[7]

References

[1] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3697–3711.

[2] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20, 1 (1960), 37–46.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In

[7] https://github.com/dbr7/RAG-metamorphic-mutation

Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). 4171–4186.

[4] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 150–158.

[5] Aoran Gan, Hao Yu, Kai Zhang, Qi Liu, Wenyu Yan, Zhenya Huang, Shiwei Tong, and Guoping Hu. 2025. Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey. *arXiv preprint arXiv:2504.14891* (2025).

[6] Jinhan Kim, Jongchan Park, and Shin Yoo. 2023. The inversive relationship between bugs and patches: An empirical study. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 314–323.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[8] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics* 12 (2024), 157–173.

[9] Joao Monteiro, Pierre-Andre Noel, Etienne Marcotte, Sai Rajeswar, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. 2024. Repliqa: A question-answering dataset for benchmarking llms on unseen reference content. *Advances in Neural Information Processing Systems* 37 (2024), 24242–24276.

[10] Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, et al. 2025. Towards trustworthy retrieval augmented generation for large language models: A survey. *arXiv preprint arXiv:2502.06872* (2025).

[11] Sachin Raja, Ajoy Mondal, and CV Jawahar. 2023. Icdar 2023 competition on visual question answering on business document images. In *International Conference on Document Analysis and Recognition*. Springer, 454–470.

[12] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems. arXiv:2311.09476 [cs.CL]

[13] Sergio Segura, Gordon Fraser, Ana B Sanchez, and Antonio Ruiz-Cortés. 2016. A survey on metamorphic testing. *IEEE Transactions on software engineering* 42, 9 (2016), 805–824.

[14] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*. PMLR, 31210–31227.

[15] Channdeth Sok, David Luz, and Yacine Haddam. 2025. MetaRAG: Metamorphic Testing for Hallucination Detection in RAG Systems. *arXiv preprint arXiv:2509.09360* (2025).

[16] Jan Strich, Enes Kutay Isgorur, Maximilian Trescher, Chris Biemann, and Martin Semmann. 2025. T2-RAGBench: Text-and-Table Benchmark for Evaluating Retrieval-Augmented Generation. *arXiv preprint arXiv:2506.12071* (2025).

[17] Victor Wang, Michael JQ Zhang, and Eunsol Choi. 2025. Improving llm-as-a-judge inference with the judgment distribution. *arXiv preprint arXiv:2503.03064* (2025).

[18] Jinyang Wu, Shuai Zhang, Feihu Che, Mingkuan Feng, Pengpeng Shao, and Jianhua Tao. 2025. Pandora's box or aladdin's lamp: A comprehensive analysis revealing the role of rag noise in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 5019–5039.

[19] Kevin Wu, Eric Wu, and James Zou. 2024. Clasheval: Quantifying the tug-of-war between an llm's internal prior and external evidence. *Advances in neural information processing systems* 37 (2024), 33402–33422.

[20] Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025* (2023).

[21] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems* 36 (2023), 46595–46623.

[22] Zexuan Zhong, Ziqing Huang, Alexander Wettig, and Danqi Chen. 2023. Poisoning retrieval corpora by injecting adversarial passages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 13764–13775.

[23] Fengbin Zhu, Wenqiang Lei, Fuli Feng, Chao Wang, Haozhou Zhang, and Tat-Seng Chua. 2022. Towards complex document understanding by discrete reasoning. In *Proceedings of the 30th ACM International Conference on Multimedia*. 4857–4866.

[24] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2025. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*. 3827–3844.

Received 2026-03-26; accepted 2026-06-18