A Resilient RAGOps Architecture for Chatbots over Knowledge Bases Derived from Multimodal Artifacts

Márcia de Sousa[1], **Gabriel Oliveira**[1], **Lucas Alves**[1], **Cleilton Rocha**[1]

[1] Instituto Atlântico

{marcia_martins,gabriel_barros,lucas_alves,cleilton_rocha}
@atlantico.com.br

Abstract. *Retrieval-Augmented Generation (RAG) chatbots rely on operational pipelines to manage knowledge in production, but building reliable knowledge bases from multimodal artifacts and recovering them after failures remain open RAGOps challenges. This paper presents a resilient RAGOps architecture for constructing and restoring such knowledge bases. It combines artifact processing, provenance tracking, embedding versioning, and rollback to transform heterogeneous artifacts into chunks and embeddings while preserving the knowledge lifecycle. Operationally instantiated, the architecture was evaluated through answer quality and resilience protocols. The stable knowledge base yielded an average BERTScore of 0.7860, which decreased to 0.7445 under the contamination scenario (approximately 5.3%) and reached 0.7945 after rollback, a level comparable to the original stable state. These results provide initial evidence of the architecture's ability to support controlled knowledge-base restoration.*

1. Introduction

Large Language Models (LLMs) have transformed the paradigm of Natural Language Processing, but they still present limitations related to factual hallucinations and the obsolescence of knowledge available after training [Serafim et al. 2025]. In this context, RAG has become a relevant strategy for grounding language models in external knowledge bases, allowing relevant documents or fragments to be retrieved at runtime and incorporated into the generation process [Lewis et al. 2021].

When RAG systems are deployed in operational applications such as chatbots, they require complex pipelines for building, updating, and querying knowledge bases [Xu et al. 2025]. This complexity increases when the knowledge base is built from multimodal artifacts, including videos, audio files, and documents, which must be converted into textual and vector representations before use [Abootorabi et al. 2025]. These pipelines include artifact ingestion, content extraction, text segmentation, embedding generation, vector indexing, and semantic retrieval. As a result, chatbot answer quality depends not only on the language model, but also on the integrity of the knowledge processing, persistence, and recovery chain [Barnett et al. 2024].

RAGOps approaches are essential for operating, managing, and maintaining RAG systems in production [Xu et al. 2025]. Despite progress in retrieval and generation, challenges remain in ensuring operational reliability. Issues such as transcription inconsistencies, processing errors, poor text segmentation, outdated embeddings, or degraded vector stores can compromise context retrieval and reduce chatbot answer quality [Barnett et al. 2024].

While many studies focus on improving retrieval metrics, model robustness, or generation quality [Salemi and Zamani 2024, Sun et al. 2024, Zhou 2024], less attention has been given to preserving and recovering the semantic integrity of the knowledge layer in multimodal RAG pipelines [Barnett et al. 2024]. Most solutions assume a standard operational flow and offer limited support for tracing relationships among artifacts, derived content, chunks, and embeddings, or for restoring validated versions of the vector store when degradation occurs [Badgujar 2026].

To address this gap, this paper proposes a resilient RAGOps architecture for RAG chatbots using knowledge bases derived from multimodal artifacts. The architecture emphasizes traceability, provenance, embedding versioning, and controlled knowledge-base recovery. It organizes the knowledge lifecycle into processing, persistence, resilience, and chatbot layers, enabling transformation of heterogeneous artifacts into chunks and embeddings, maintenance of provenance relationships, and restoration of validated vector representations through rollback. This approach supports multimodal artifact ingestion and helps mitigate vector store degradation without requiring full pipeline reprocessing.

The proposed architecture was implemented as an operational pipeline and evaluated using two complementary protocols. The first assessed answer quality from a stable knowledge base. The second tested restoration capability after controlled contamination, comparing stable, contaminated, and recovered states. Results provide initial evidence supporting both constructing knowledge bases from multimodal artifacts and recovering answer quality after degradation.

2. Background and Related Work

2.1. Software Resilience

Software resilience is defined by Meyer [Meyer 2009] as the ability of a system to adapt in order to absorb and tolerate failures, attacks, and environmental changes while preserving its functional and non-functional characteristics. Unlike robustness, resilience is treated as a composite property that supports the analysis of a system's adaptive capacity. In this sense, De Florio [De Florio 2013] structures resilience into four attributes: perception, awareness, planning, and dynamicity. These attributes respectively represent the detection of changes, the understanding of their impacts, the definition of adaptive actions, and the extent to which such capabilities are designed or emerge at runtime.

In the context of data and AI pipelines, resilience has been explored mainly through mechanisms such as versioning, historical recovery, and operational self-healing. Prajapati [Prajapati 2025], for example, proposes *Live VectorLake*, a temporal infrastructure based on incremental versioning and *Point-in-Time Retrieval* (PITR). Badgujar [Badgujar 2026], in turn, discusses *Self-Healing Data Pipelines*, in which anomaly detection mechanisms trigger corrective actions such as *retry*, *rollback*, and policy-driven remediation. These approaches highlight the importance of preserving historical states, detecting degradation, and restoring reliable versions of data and pipelines in operational environments.

Although these works contribute to data recovery and operational workflows, they do not directly address the semantic integrity of vector stores in multimodal RAG pipelines. This gap is particularly relevant in RAG systems, where answer quality depends not only on the language model, but also on the availability, integrity, and freshness

of external knowledge bases. Failures in vector databases, ingestion pipelines, or indexing processes may compromise context retrieval and, consequently, the accuracy of the generated answers. Therefore, the operation of RAG systems requires mechanisms that can detect degradation, preserve traceability, and restore the knowledge base to a consistent state, thereby motivating the discussion of *RAGOps*.

2.2. RAGOps

The operationalization of RAG systems in production environments has recently been discussed under the concept of *RAGOps*. Xu et al. [Xu et al. 2025] formalize the term as a specialized extension of *LLMOps*, focusing on governance, automation, continuous evaluation, and lifecycle management of dynamic data in RAG pipelines. This perspective provides an important conceptual basis for understanding the production challenges of these systems, but remains predominantly at a taxonomic and architectural level.

Additional research addresses complementary aspects of reliability in RAG systems. Badillo Rangel [Badillo-Rangel 2024] integrates RAG, *Parameter-Efficient Fine-Tuning*, and the Medallion Architecture to structure the data lifecycle and assess answers using metrics such as BLEU, ROUGE, and BERTScore. Tu et al. [Tu et al. 2025] introduce *Robust Fine-Tuning* (RbFT), which enhances the language model's resilience to irrelevant, noisy, or counterfactual documents retrieved during inference. Jiao et al. [Jiao et al. 2026] propose *Doctor-RAG*, a framework for diagnosing and repairing failures in *Agentic RAG* architectures. This approach localizes errors across retrieval and reasoning stages, categorizes failures, and implements targeted correction strategies, reusing previously validated segments of the inference trajectory to minimize the cost of full re-executions.

Despite these advancements, current approaches typically address isolated layers of the problem: some prioritize governance and operation of RAG pipelines, others enhance the robustness of the language model, and some focus on failure diagnosis during inference. In contrast, the present work addresses both the construction of RAG knowledge bases from multimodal artifacts and the operational resilience of the RAG infrastructure. To this end, it treats the vector store and its *embeddings* as versioned, traceable, and recoverable components. Accordingly, this study addresses the challenge of preserving and restoring the semantic integrity of the knowledge layer throughout its lifecycle, thereby contributing to a more resilient *RAGOps* architecture for knowledge bases derived from multimodal artifacts.

3. Conceptual Architecture and Instantiation

The proposed architecture defines the operational lifecycle of a RAG knowledge base built from multimodal artifacts and consumed by a chatbot. It organizes knowledge construction, persistence, recovery, retrieval, and answer generation across four layers: *Data Processing*, *Persistence*, *Resilience*, and *Chatbot*, as shown in Figure 1. These layers transform multimodal artifacts into textual chunks and embeddings, preserve provenance records, restore affected *embeddings*, and support grounded answer generation. In the instantiation, *Apache Airflow* orchestrates coordinated workflows that operationalize the components assigned to each architectural layer.

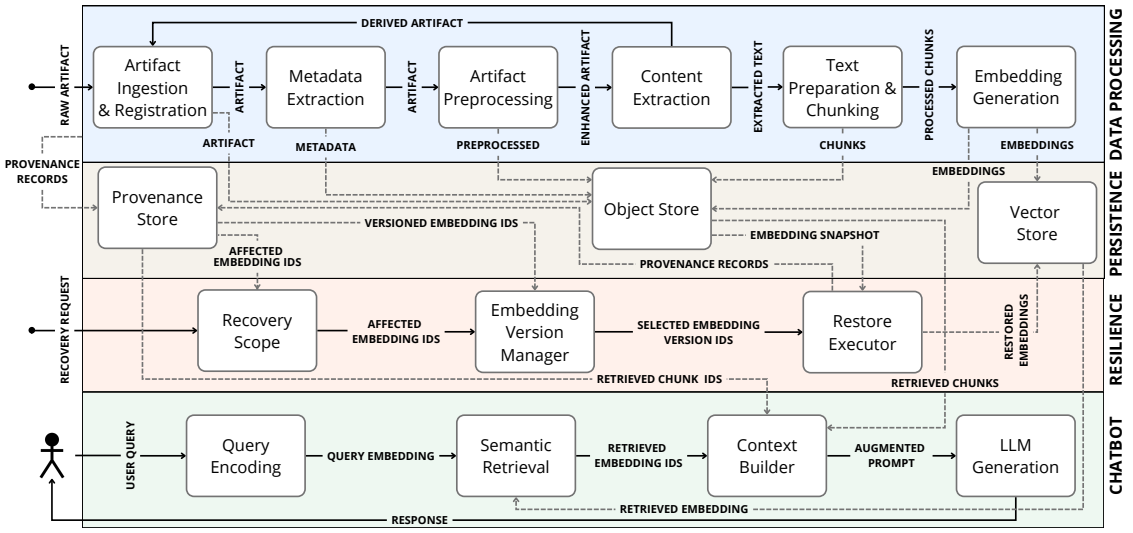


Figure 1. Conceptual resilient RAGOps architecture for chatbots over knowledge bases derived from multimodal artifacts.

3.1. Data Processing Layer

The *Data Processing* layer transforms raw multimodal artifacts, such as videos, audio files, and documents, into fragmented and vectorized textual representations for use by the Chatbot layer. As shown in Figure 1, the workflow begins with artifact ingestion and registration, followed by metadata extraction, preprocessing, and content extraction to obtain a textual representation. The resulting text is then prepared, segmented into *chunks*, and used to generate *embeddings*. All produced artifacts, metadata, provenance records, *chunks*, and *embeddings* are sent to the persistence layer, ensuring traceability and supporting querying and recovery.

The *Artifact Ingestion and Registration* component receives the raw artifact, records its entry, and assigns an identifier, source, and initial processing context. This step ensures the artifact can be traced, reprocessed, or linked to derived data. The *Metadata Extraction* component then collects descriptive and technical information, such as size, duration, format, language, word count, and other media-specific attributes. These metadata support auditing, filtering, traceability, and recovery. The *Artifact Preprocessing* component then applies necessary transformations, including format normalization, cleaning, quality enhancement, signal preparation, or other adjustments based on artifact type.

The *Content Extraction* component extracts a content representation from the preprocessed artifact suitable for the next stages. This representation depends on the artifact type: for videos, it may involve audio extraction; for audio files, transcription; and for documents, direct text extraction. A key aspect of this component is its reentrant behaviour. If the extracted content does not constitute a textual representation, the derived artefact is reintroduced into the pipeline and processed repeatedly until text suitable for RAG preparation is obtained.

After obtaining a textual representation, the content is sent to the *Text Preparation and Chunking* component, which normalizes, cleans, and segments the text into seman-

tically coherent *chunks*. These *chunks* serve as the basic knowledge units for *embedding* generation and semantic retrieval. The *Embedding Generation* component converts the processed *chunks* into vector representations. Each *embedding* semantically represents a text fragment and enables the chatbot layer to perform similarity searches. The *Data Processing* layer therefore produces both chunks and vector representations from text required by the *Chatbot* layer for semantic retrieval.

In the implementation, this layer follows the *Medallion Architecture* (Bronze, Silver, and Gold) [Steelman Jr 2025]. Although the conceptual architecture supports heterogeneous artifacts, the current instantiation focuses on videos; thus, multimodality concerns source ingestion and processing, while retrieval and answer generation rely only on derived textual representations, with visual features such as *thumbnails* not used in embeddings or generation. Videos are ingested, preprocessed, and decomposed into audio and *thumbnails*. The audio is transcribed with *Whisper*, refined, translated into English using *deep-translator*, semantically chunked, and embedded using the *all-MiniLM-L6-v2* model.

3.2. Persistence Layer

The *Persistence* layer supports the workflow by storing artifacts and records generated during the *Data Processing* layer. The *Object Store* retains both raw and derived artifacts, including original files, videos, audio files, transcriptions, metadata, preprocessed versions, extracted texts, generated *chunks*, and historical *embeddings*. It maintains artifact versions to enable auditing, reprocessing, and recovery of previous states.

The *Provenance Store* maintains provenance records generated throughout the data processing layer. Unlike the *Object Store*, it does not store content, but records relationships among artifacts, extracted content, *chunks*, and *embeddings*. In the current implementation, embedding states are versioned, while chunks are persisted in the Object Store and linked through provenance records to their corresponding embedding versions, preserving their processing context. The chunking strategy remains fixed in the current evaluation. This enables tracing artifact transformations and supports selective recovery of *embeddings* while preserving their relationships with the corresponding *chunks*. The *Vector Store* contains active *embeddings* generated from processed *chunks* and makes them available for semantic retrieval, representing the latest knowledge-base information.

In this architecture, the persistence layer integrates three main technologies. *PostgreSQL* implements the *Provenance Store*, centralizing identifiers, version histories, and provenance relationships among artifacts, transcriptions, *chunks*, and *embeddings*. *MinIO* serves as the *Object Store*, preserving raw artifacts and intermediate versions, including videos, metadata, extracted audio files, transcriptions, and other derived objects. *Milvus* implements the *Vector Store*, indexing and managing active *embeddings* generated from *chunks*.

3.3. Resilience Layer

The *Resilience* layer coordinates recovery mechanisms when failures, inconsistencies, or restoration needs impact the knowledge base's vector representation. The *Persistence* layer stores artifacts, active *embeddings*, versions, and provenance records, which the *Resilience* layer uses to restore a consistent state. Its primary goal is to enable targeted recovery of affected *embeddings* without reprocessing all ingested artifacts.

The recovery process begins with a *Recovery Request*, which specifies the affected artifact. This request may be issued after a degradation, inconsistency, or validation failure is associated with an artifact previously registered in the pipeline. The *Recovery Scope* component then identifies which artifact's vector representation requires restoration. It queries the *Provenance Store* to relate the artifact to its extracted content, *chunks*, and corresponding *embeddings*, determining the set of *embeddings* associated with the affected artifact. Once the scope is defined, the identifiers of the affected *embeddings* are sent to the *Embedding Version Manager*. This component manages available *embedding* versions and selects those needed to restore the desired vector store state. It uses versioned identifiers and provenance records to ensure recovery considers both the vectors and their relationships with artifacts, *chunks*, and processing context.

The *Restore Executor* then performs the restoration. Using the selected version identifiers, it retrieves *embedding* snapshots from the *Object Store* and reinserts them into the *Vector Store*. This restores the previously validated vector representation for the artifact. The workflow enables selective recovery of *embeddings* without full reprocessing and preserves traceability between the recovered state, provenance records, and original artifacts.

This layer is implemented as a recovery *workflow* in *Airflow*. The *Recovery Request* acts as a trigger in the orchestrator, receiving the affected artifact's identifier and a temporal marker for a stable state. The *Embedding Version Manager* uses *Feast* as a *Feature Store* to perform *Point-in-Time Recovery* based on history in *PostgreSQL*. The *Restore Executor*, developed in *Python* as an *Airflow* task, removes compromised vectors from *Milvus*, validates recovered data, and reinserts restored *embeddings* into the vector database.

In the current instantiation, degradation detection and recovery-policy selection are outside the evaluation scope. The recovery workflow is explicitly triggered with the affected artifact and a temporal marker corresponding to a previously validated state.

3.4. Chatbot Layer

The *Chatbot* layer uses the knowledge base maintained by previous layers to answer user queries. It does not process raw multimodal artifacts directly; instead, it uses active *embeddings* from processed artifacts to retrieve the most relevant *chunks* for each query. This ensures user interaction occurs over a structured, vectorized, and traceable knowledge base.

The workflow begins when a user submits a query to the *Query Encoding* component, which transforms the question into a *query embedding* compatible with the *Vector Store*. The *Semantic Retrieval* component then searches the *Vector Store* for *embeddings* that are semantically closest to the query. This process returns a set of embedding identifiers representing the most relevant knowledge fragments for answering the user's question. Next, the *Context Builder* uses the retrieved embedding identifiers to obtain the corresponding *chunks* and related information. It may query provenance records to map *embeddings* to *chunks* and source artifacts, retrieving the content needed to build the answer context. This step is essential because the language model requires an organized textual context, not the raw vectors.

Finally, the *LLM Generation* component receives the *augmented prompt*, which

includes the user query and retrieved context, and generates the final answer. This answer is grounded in the *chunks* selected through semantic search, ensuring responses align with the knowledge base. The chatbot layer thus completes the architecture cycle by using active *embeddings*, provenance records, and retrieved content to provide answers based on the most current and consistent knowledge base state.

This layer is implemented as an *Airflow* workflow, with semantic retrieval performed in *Milvus* using cosine similarity. The *Context Builder* retrieves chunks and provenance metadata from *PostgreSQL*, while LLM Generation uses *Ollama* and *Llama* 3 to generate answers from the augmented prompt.

Together, these layers support the construction, preservation, recovery, and use of traceable knowledge bases derived from multimodal artifacts. Their separation of responsibilities enhances traceability, maintainability, and controlled recovery in RAGOps environments.

4. Evaluation

The evaluation followed a black-box approach, inspired by [Li et al. 2025], analyzing the architecture end-to-end through its observable inputs and outputs. This method suits our context because the solution integrates processing, persistence, recovery, and generation, and its effectiveness depends on the interaction among these components. As shown in Figure 2, the evaluation consisted of three phases: constructing the *Ground Truth* from reference videos and transcriptions; preparing the stable, contaminated, and recovered knowledge bases; and executing the evaluation protocols. In these protocols, the chatbot generates answers using the prepared knowledge bases, and these answers are compared against the expected answers from the *Ground Truth*. All experiments used the architecture instance described in Section 3. The evaluation was designed as an initial proof-of-concept assessment of the architecture's feasibility rather than as a comprehensive validation of its generalizability across large-scale RAG workloads.

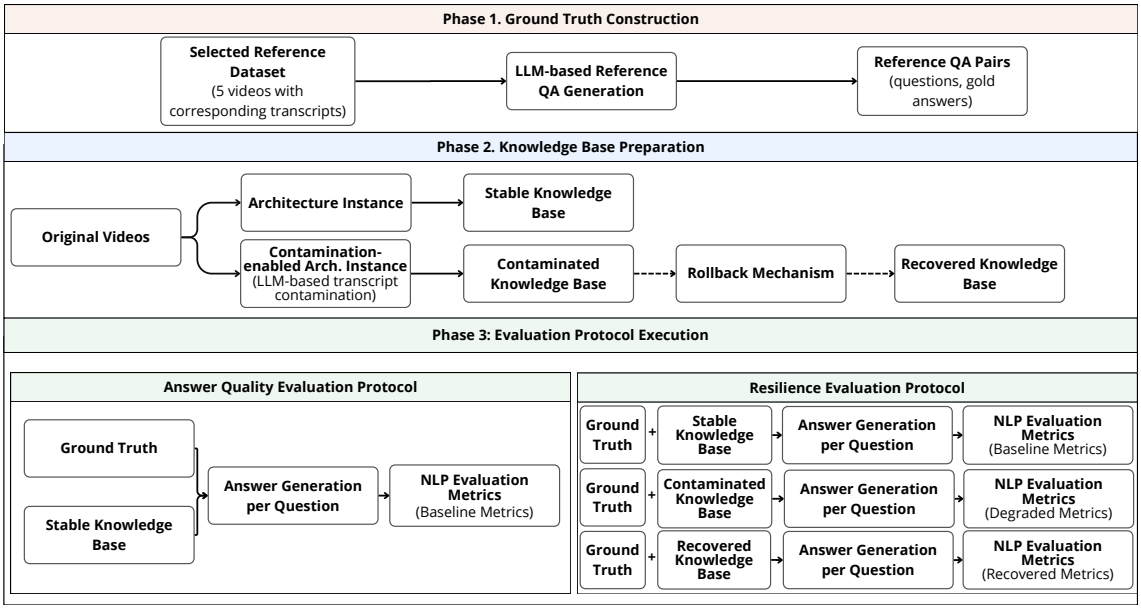


Figure 2. Overview of the proposed evaluation methodology.

4.1. Ground Truth Construction

To construct the *Ground Truth*, we selected five video samples from the public repository *Video And Transcript Dataset*[1]. The videos, in English and lasting 30 seconds to 1 minute and 20 seconds, include human-reviewed transcriptions. Using audiovisual artifacts allows us to test the architecture's main workflows, from video processing to knowledge base construction for chatbot queries.

Using the transcriptions, we generated synthetic question-answer pairs with the *LLM-based Reference QA Generation* component and the Gemini 3.5 Flash model[2]. Each transcription produced three distinct pairs, totaling 15 evaluation pairs. These *Reference QA Pairs* associate each question with an expected, or *gold*, answer. We limited the pairs to facts explicitly present in the transcriptions and manually verified that the questions were answerable and the expected answers aligned with the reference content.

4.2. Knowledge Base Preparation

The second phase prepares the knowledge bases for the evaluation protocols. First, the *Original Videos* are processed by the *Architecture Instance*, which extracts and prepares textual content, produces *chunks*, and generates *embeddings* to form the *Stable Knowledge Base*. This base represents the reliable state of the solution and serves as both the reference for answer quality and the baseline for resilience evaluation.

Next, the same videos are processed by a *Contamination-enabled Architecture Instance*. Here, the generated transcription is intercepted before *chunking* and vectorization, then modified by Gemini model to simulate semantic and syntactic noise. This produces the *Contaminated Knowledge Base*, representing a controlled degradation scenario. The *Rollback Mechanism* then restores the previously validated *embeddings*, resulting in the *Recovered Knowledge Base*.

4.3. Evaluation Protocols

The evaluation used two complementary protocols. The first, the *Answer Quality Evaluation Protocol*, assesses answer quality when the knowledge base is stable. Questions from the *Ground Truth* are submitted to the answer generation component using the *Stable Knowledge Base*. Each generated answer is compared to its *gold answer* using ROUGE-1, ROUGE-2, ROUGE-L, BLEU, METEOR, and BERTScore metrics to assess both lexical overlap and semantic similarity [Lin 2004, Papineni et al. 2002, Lavie et al. 2004, Zhang* et al. 2020]. To minimize the impact of random variations, each question was run three times independently.

The second protocol, the *Resilience Evaluation Protocol*, measures the architecture's ability to restore answer quality after controlled degradation of the knowledge base. The same *Ground Truth* questions are run on three states: *Stable Knowledge Base*, *Contaminated Knowledge Base*, and *Recovered Knowledge Base*. This comparison reveals the impact of contamination and whether the *Rollback Mechanism* restores answer quality to near-stable levels. A decrease in metrics in the contaminated state shows degradation, while metric recovery after rollback demonstrates that versioning and restoration help reestablish semantic quality without full reprocessing.

[1] `https://www.kaggle.com/datasets/dogepuresi/videoandtranscriptdataset`
[2] `https://deepmind.google/models/gemini/flash/`

5. Results and Discussion

This section discusses the results obtained from the two evaluation protocols: the *Answer Quality Evaluation Protocol*, which focuses on the analysis of the answers generated using the stable knowledge base, and the *Resilience Evaluation Protocol*, which focuses on the analysis of knowledge-base degradation and recovery. The *Reference Dataset*, the *Reference QA Pairs*, the *prompts*, and the remaining artifacts generated during the evaluation are available in the study repository[3].

5.1. Answer Quality Results

Table 1 presents the results obtained with the *Stable Knowledge Base*. For each video, three questions from the *ground truth* were considered, and each question was executed three times, resulting in nine answers per video. The reported values correspond to the arithmetic mean of the metrics calculated for these executions.

Table 1. Average answer-quality metrics using the *Stable Knowledge Base*.

Video	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	METEOR	BERTScore
V1	0.5129	0.3951	0.4123	0.1574	0.5055	0.7958
V2	0.4267	0.2128	0.3682	0.0545	0.4734	0.7865
V3	0.3146	0.3713	0.2534	0.7440	0.3517	0.7383
V4	0.5388	0.4353	0.5161	0.4184	0.6592	0.8303
V5	0.4751	0.3477	0.4368	0.3193	0.6138	0.7790

Overall, the results indicate that the architecture was able to produce answers with reasonable alignment with the reference answers. The overall average BERTScore was approximately 0.7860, suggesting that the generated answers preserved part of the expected semantic content, even when they did not reproduce the same textual formulation as the *gold answers*. This interpretation is also supported by the overall average METEOR score, approximately 0.5207, which indicates moderate correspondence between the generated and expected answers.

The difference between lexical and semantic metrics reinforces this interpretation. While lexical metrics varied more across the videos, BERTScore remained relatively stable. This behavior is expected in generative systems, in which an answer may preserve the expected meaning even when using terms or structures that differ from the reference. Therefore, lexical metrics are useful for observing textual overlap, but should be interpreted together with semantic metrics.

The main exception observed was video V3, which presented the lowest relative performance in most metrics. This case suggests that the final answer quality may vary according to the interaction among semantic retrieval, context construction, and text generation, as well as artifact-specific characteristics such as transcription length and content specificity. Since the evaluation was not designed to isolate the contribution of these factors, no single cause can be attributed to the observed behavior.

5.2. Resilience Results

The resilience evaluation uses the results obtained with the *Stable Knowledge Base* as a baseline. From this validated state, the base evolves to a contaminated version, making it

[3] https://github.com/lucasvieira123/Resilient_RAGOps

possible to observe the impact of degradation on the answers. Then, the *Rollback Mechanism* restores a previous state of the base, producing the *Recovered Knowledge Base*. Thus, the analysis considers three observable states: the stable state, previously reported in Table 1, and the contaminated and recovered states, presented in Table 2.

In aggregate terms, contamination reduced answer quality: the average BERTScore decreased from approximately 0.7860 in the stable state to 0.7445 in the contaminated state, representing a reduction of about 5.27% in semantic similarity. This decrease suggests that the degradation introduced into the knowledge base affected the semantic alignment between the generated answers and the *gold answers*.

Table 2. Average answer-quality metrics after contamination and recovery.

Video	State	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	METEOR	BERTScore
V1	Contaminated	0.2784	0.1403	0.2086	0.0498	0.3948	0.7187
	Recovered	0.4288	0.2451	0.3404	0.0661	0.4307	0.7823
V2	Contaminated	0.3664	0.1718	0.3036	0.0000	0.3271	0.7646
	Recovered	0.4788	0.2866	0.4135	0.1109	0.5491	0.8039
V3	Contaminated	0.3701	0.2041	0.2806	0.0738	0.3921	0.7563
	Recovered	0.3448	0.1874	0.2973	0.0859	0.4106	0.7576
V4	Contaminated	0.3610	0.2006	0.3262	0.0997	0.5513	0.7677
	Recovered	0.5709	0.4778	0.5608	0.3569	0.6460	0.8395
V5	Contaminated	0.2386	0.0715	0.1867	0.0119	0.2975	0.7152
	Recovered	0.5350	0.3810	0.4842	0.1924	0.6422	0.7891

After applying the *Rollback Mechanism*, the metrics showed an overall improvement compared with the contaminated scenario. The average BERTScore increased to approximately 0.7945, reaching a level comparable to that observed with the *Stable Knowledge Base*. This result suggests that restoring the versioned *embeddings* helped recover the semantic alignment between the generated answers and the *gold answers*.

Recovery is also observed in the lexical metrics, although their values vary more across videos. In general, ROUGE, BLEU, and METEOR follow the trend of decreasing in the contaminated scenario and improving after the *rollback*, indicating that recovery does not occur only at the semantic level, but also in the textual approximation between the generated answers and the *gold answers*.

Despite the overall recovery trend, the effect of the *rollback* was not uniform across all cases. Video V3 was the main exception: after recovery, its BERTScore remained below the stable scenario and close to the contaminated scenario. However, ROUGE-L, BLEU, and METEOR exceeded the values obtained with the contaminated data, suggesting partial recovery in lexical correspondence. This behavior may be related to the natural variability of LLM-based generation and to the sensitivity of automatic metrics to small changes in the retrieved context or generated wording. Therefore, the results should not be interpreted as evidence that *rollback* improves the original base. Rather, they suggest that, on average, rollback can restore answer quality to a level comparable to the previous stable state.

Overall, the results provide initial evidence that embedding versioning and *rollback* can support controlled recovery of knowledge-base quality after degradation. However, as this initial feasibility study used five short videos, 15 reference question-answer pairs, and a single controlled contamination scenario, the findings should be interpreted as an initial validation rather than evidence of generalization to larger knowledge bases

or diverse failure conditions. The evaluation also does not establish superiority over conventional database backup and restoration approaches, which would require further comparison regarding restoration granularity, provenance preservation, reprocessing cost, and recovery time.

6. Conclusion

This work presented a resilient RAGOps architecture for RAG chatbots supported by knowledge bases derived from multimodal artifacts, oriented toward traceability, versioning, and recovery of the knowledge base used by RAG systems. The proposed architecture organizes the knowledge lifecycle into processing, persistence, resilience, and chatbot layers, enabling artifacts, chunks, embeddings, and provenance records to be maintained and restored in a controlled manner. The evaluation provided initial evidence of the feasibility of the proposed solution in two main aspects. First, the architecture was able to build a stable knowledge base from audiovisual artifacts and use it to generate answers aligned with the reference content. Second, under knowledge-base contamination, the rollback mechanism restored a previous knowledge state, mitigating the effects of degradation without requiring the complete reprocessing of the pipeline. Quantitatively, the average BERTScore decreased from 0.7860 in the stable state to 0.7445 after contamination and reached 0.7945 after rollback, returning to a level comparable to the original stable state. As future work, we intend to extend the evaluation to other artifact types and larger knowledge bases, explore additional failure conditions and chunking configurations, compare the restoration mechanism with conventional backup strategies, and investigate automatic mechanisms for detecting degradation and triggering rollback.

References

Abootorabi, M. M., Zobeiri, A., Dehghani, M., Mohammadkhani, M., Mohammadi, B., Ghahroodi, O., Baghshah, M. S., and Asgari, E. (2025). Ask in any modality: A comprehensive survey on multimodal retrieval-augmented generation. *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16776–16809.

Badgujar, P. (2026). Self-healing data pipelines: Leveraging ai to detect and correct failures in real-time. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 7(1):87–90.

Badillo-Rangel, E. (2024). Enhancing llm performance in specialized spanish domains using rag and peft qlora.

Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., and Abdelrazek, M. (2024). Seven failure points when engineering a retrieval augmented generation system.

De Florio, V. (2013). On the constituent attributes of software and organizational resilience. *Interdisciplinary Science Reviews*, 38(2):122–148.

Jiao, S., Huang, C., Qi, S., Wang, X., Li, Y., and Yao, L. (2026). Doctor-rag: Failure-aware repair for agentic retrieval-augmented generation. *arXiv preprint arXiv:2604.00865*.

Lavie, A., Sagae, K., and Jayaraman, S. (2004). The significance of recall in automatic metrics for mt evaluation. In *Conference of the Association for Machine Translation in the Americas*, pages 134–143. Springer.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive nlp tasks.

Li, S., Zhang, H., Wang, Y., Zhou, M., Ma, Z., and Liang, F. (2025). Universal end-to-end automated evaluation framework for retrieval-augmented generation system. In *2025 IEEE 8th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 867–871. IEEE.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Meyer, J. F. (2009). Defining and evaluating resilience: A performability perspective. In *Proceedings of the International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS)*, volume 1, page 561.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Prajapati, T. (2025). Livevectorlake: A real-time versioned knowledge base architecture for streaming vector updates and temporal retrieval.

Salemi, A. and Zamani, H. (2024). Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2395–2400.

Serafim, L. M. et al. (2025). Do humano aos llms: Apontamentos sobre o processamento de linguagem.

Steelman Jr, R. L. (2025). Medallion architecture. In *Mastering Snowflake DataOps with DataOps. live: An End-to-End Guide to Modern Data Management*, pages 247–264. Springer.

Sun, T., Somalwar, A., and Chan, H. (2024). Multimodal retrieval augmented generation evaluation benchmark. In *2024 Ieee 99th vehicular technology conference (vtc2024-Spring)*, pages 1–5. IEEE.

Tu, Y., Su, W., Zhou, Y., Liu, Y., and Ai, Q. (2025). Robust fine-tuning for retrieval augmented generation against retrieval defects. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 1272–1282, New York, NY, USA. Association for Computing Machinery.

Xu, X., Weytjens, H., Zhang, D., Lu, Q., Weber, I., and Zhu, L. (2025). Ragops: Operating and managing retrieval-augmented generation pipelines.

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhou, R. (2024). Advanced embedding techniques in multimodal retrieval augmented generation a comprehensive study on cross modal ai applications. *Journal of Computing and Electronic Information Management*, 13(3):16–22.