

RAGtio: A Modular Framework for Systematic Evaluation of Hybrid Retrieval-Augmented Generation Pipelines

Annamaria Defilippo

University of Catanzaro

Nicola Procopio

Technology Innovation HUB

Pietro Hiram Guzzi

hguzzi@unicz.it

University of Catanzaro

Pierangelo Veltri

University of Calabria

Patrizia Vizza

University of Calabria

software

Keywords:

Posted Date: August 13th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10503006/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

RAGtio: A Modular Framework for Systematic Evaluation of Hybrid Retrieval-Augmented Generation Pipelines

Annamaria Defilippo¹, Nicola Procopio², Pietro Hiram Guzzi^{1,*},
Pierangelo Veltri³, Patrizia Vizza³

¹Department of Surgical and Medical Sciences, Magna Graecia University of Catanzaro, Catanzaro, Italy

²Technology Innovation Hub, AC Software, Lamezia Terme, Italy

³DIMES, University of Calabria, Rende, Italy

{annamaria.defilippo,hguzzi}@unicz.it

nicola.procopio@acsoftware.it

{patrizia.vizza,pierangelo.veltri}@dimes.unical.it

*Corresponding author: Pietro Hiram Guzzi; email: hguzzi@unicz.it

Abstract

Retrieval-Augmented Generation (RAG) architectures combine Large Language Models (LLMs) with external knowledge stores to produce grounded, verifiable answers. While RAG systems have been increasingly adopted in the biomedical domain, existing work focuses primarily on downstream question-answering (QA) performance, with limited attention to systematic evaluation and usability for non-technical users. This paper presents RAGtio, a modular, open-source framework for the rigorous and reproducible evaluation of RAG retrieval pipelines over domain-specific text corpora, with primary application to biomedical documents. The framework implements and allows the comparison of four retrieval configurations: sparse, dense, hybrid, and hybrid with cross-encoder re-ranking, using a weighted Reciprocal Rank Fusion (RRF) mechanism. A key contribution is the introduction of two complementary evaluation modes: (i) Mode A automatically generates synthetic queries from randomly sampled document chunks via an LLM, enabling rapid large-scale assessment; (ii) Mode B accepts user-provided query relevance pairs assembled from domain expertise, yielding an unbiased estimate of retrieval quality on genuinely novel queries. The framework is built on the Haystack library and Qdrant vector database, supports heterogeneous document formats, and exposes a YAML-based configuration interface together with a web-based graphical interface for running evaluations and inspecting results without software development expertise.

1 Introduction

Large Language Models (LLMs) have enabled impressive progress in natural language understanding and generation [6, 2], but they still suffer from limitations such as hallucinations, outdated knowledge, and difficulty in providing verifiable references to source material [23]. Retrieval-Augmented Generation (RAG) addresses these issues by coupling LLMs with external document stores, retrieving relevant content at query time and injecting it into the model’s context [18, 13, 28]. RAG systems are especially attractive for scenarios where document collections are frequently updated, domain-specific, or not available during LLM pre-training [4].

Designing the retrieval component is a central problem in the engineering of RAG systems [4]. *Dense retrievers* map documents and queries into a shared vector space and rank candidates by semantic similarity, while *sparse retrievers* such as BM25 exploit exact term matches and inverse document frequency statistics [26, 25]. Dense approaches generally capture paraphrases and implicit semantics but may underperform with highly specialised or low-frequency terminology, whereas sparse approaches handle rare tokens well but are sensitive to vocabulary mismatches and phrasing variations. In practice, many applications require retrieval

solutions capable of handling heterogeneous query styles without extensive domain-specific fine-tuning [29]. To address these complementary limitations, hybrid retrieval frameworks combine dense and sparse strategies and can further incorporate cross-encoder re-rankers for additional precision refinement.

Recent surveys highlight the rapid expansion of RAG architectures, including modular pipelines, and domain-specific applications in healthcare and biomedicine [13, 27, 1, 3, 19]. Within this evolving landscape, hybrid retrieval approaches have emerged as particularly effective, with several studies reporting consistent improvements from fusion-based strategies such as Reciprocal Rank Fusion (RRF), especially in complex and specialised corpora [7, 20, 15].

Despite this growing body of work, the evaluation of retrieval components remains far from standardised, particularly in the biomedical domain. Most studies primarily focus on end-to-end QA accuracy, making it difficult to isolate the contribution of the retrieval stage itself, while retrieval-specific metrics are often applied inconsistently or without reproducible evaluation protocols [5, 14]. This limitation is especially relevant in biomedical settings, where retrieval quality directly affects factual grounding, traceability, and reproducibility, yet evaluation pipelines are frequently implemented on an ad hoc basis [1, 19].

This paper presents RAGtio, a modular and domain-agnostic framework for the systematic evaluation of RAG retrieval pipelines. RAGtio generalises the retrieval workflow of our previous domain-specific system into a configurable and explicitly evaluable platform [10]. The framework enables controlled comparisons across retrieval configurations, reports four complementary information retrieval (IR) metrics, and separates synthetic evaluation queries from independently curated query–relevance pairs.

In addition to its methodological contribution, RAGtio is provided as operational research software rather than solely as a conceptual architecture. The complete implementation is publicly available at <https://github.com/ArchAI-Labs/ragtio> and can be deployed through Docker Compose. Its services integrate Haystack [9], Qdrant, local LLM inference, a REST API, and a browser-based graphical interface. The repository includes the source code and configuration resources required to reproduce and adapt the workflow, making the framework directly inspectable, deployable, and extensible by other researchers.

Practical accessibility is addressed at both configuration and interaction levels. Retrieval, chunking, embedding, reranking, and evaluation settings can be changed through YAML files without modifying the source code. Through the web interface, users can upload and index documents, run both evaluation modes, inspect per-query results, curate relevance judgements, and test retrieval and answer generation interactively. The case study across four biomedical domains demonstrates that these functions operate within a complete workflow, from corpus ingestion to result inspection. This constitutes evidence of functional availability and practical usability; however, the study does not claim a formal human–computer interaction evaluation, which would require user testing with predefined usability instruments and a larger participant sample.

The main contributions of this work are:

- A modular framework covering document ingestion, indexing, retrieval, answer generation, and retrieval-focused evaluation.
- Configurable sparse, dense, hybrid, and reranked hybrid retrieval strategies assessed through four standard IR metrics.
- Two complementary evaluation modes based on synthetic and human-curated queries, enabling users to examine sensitivity to query-generation bias.
- A publicly available, containerised implementation with YAML configuration, a REST API, and a web interface that supports reproducible workflows without requiring users to implement pipeline components programmatically.

Overall, RAGtio combines retrieval-evaluation methodology with an effectively available software implementation. Its open repository, containerised deployment, configuration interface, and interactive web workflow are intended to make systematic RAG evaluation more transparent, reproducible, and accessible, particularly in specialised domains such as biomedicine.

2 Related Work

RAG systems extend LLMs by retrieving external knowledge during inference, typically from document stores, vector databases, or knowledge graphs [18, 13]. Existing frameworks differ in retrieval strategy, indexing granularity, grounding mechanisms, and evaluation protocols [33]. In fact, some approaches rely primarily on dense retrieval with bi-encoders and domain-specific pre-training, while others emphasise sparse or hybrid retrieval, graph-based reasoning, or modular pipelines with dedicated retrievers, rankers, and memory components [31, 21]. Four complementary axes are discussed in this section: retrieval strategies underlying modern RAG pipelines, domain-specific systems developed for biomedical and health informatics applications, evaluation practices adopted to assess retrieval quality, and the broader ecosystem of frameworks and tooling supporting reproducibility. Table 1 summarises how representative domain-specific systems compare along these dimensions.

Retrieval strategies. Sparse retrievers rank documents based on lexical matching between query and document terms, typically using probabilistic retrieval models such as BM25. They are particularly effective for specialised terminology and exact keyword matching but may fail to capture semantic similarity when relevant documents use different wording. Dense retrievers map documents and queries into a shared embedding space, capturing semantic similarity and paraphrase relations, but may underperform on specialised or low-frequency vocabulary [26]. Hybrid architectures that combine lexical signals from sparse retrieval and semantic signals from dense retrieval have been widely studied, including RRF-based approaches, score interpolation, and learned ensembles [7, 26, 20, 15]. A cross-encoder re-ranker applied to the initial retrieval pool can further improve precision by scoring query-document pairs jointly, at the cost of additional latency [22, 12, 27]. Building on these foundations, this work integrates all four configurations (sparse, dense, hybrid, and hybrid with re-ranking) into a single, switchable pipeline that can be selected at runtime through configuration rather than code changes. This design enables the four strategies to be compared directly against one another under identical conditions, unlike prior comparative studies that benchmark individual techniques against heterogeneous baselines drawn from different corpora, pre-processing pipelines, or evaluation protocols [29]. The proposed platform allows an internal comparison within the framework rather than against external systems, and does not rely on domain-specific fine-tuning of any component, isolating the effect of the retrieval strategy itself from confounding factors such as corpus composition or model adaptation.

Domain-specific RAG systems. In the biomedical and health informatics domain, RAG-based systems have been proposed for QA, clinical trial screening, and literature exploration, often built around specialised corpora and benchmarks [33, 30, 17, 8, 3, 19]. As summarised in Table 1, these systems advance the state of the art in their respective tasks; however, their designs are typically tightly coupled to specific corpora, languages, or evaluation settings, and few provide reusable, configurable software artefacts [17, 8]. The present work takes a complementary direction: rather than optimising for a single domain, it prioritises modularity, configurability, and accessibility, supporting heterogeneous document formats (i.e., plain text, Markdown, PDF, DOCX, CSV, JSON/JSONL), pluggable embedding models, and runtime reconfiguration without service restart.

Table 1 makes this pattern explicit across six representative systems. KRA-GEN [21] focuses on knowledge-grounded generation over curated biomedical corpora and does not expose a configurable retrieval pipeline, while BAILICAI [8] integrates cross-encoder re-ranking but reports retrieval evaluation only within its case study, without releasing reusable benchmarking tools. Similarly, TC-RAG [17] includes retrieval metrics in its clinical-trial analysis but does not provide a general-purpose retrieval harness applicable to external corpora. MIRAGE [33] offers a comprehensive retrieval evaluation protocol and reusable benchmarking tools, yet it does not include configurable retrieval pipelines or re-ranking components. Self-BioRAG [16] and MedRAG [34] incorporate re-ranking and report retrieval performance within their experimental setups, but both pipelines remain tightly coupled to domain-specific corpora and lack modular, reusable retrieval configurations.

RAG evaluation and self-referential bias. Most existing studies report end-to-end QA accuracy, making it difficult to isolate the contribution of the retrieval stage [33]. Self-referential bias is particularly acute

Table 1: Comparison of representative RAG systems and frameworks along four dimensions.

Configurable Retrieval Pipeline indicates whether the retrieval strategy (e.g., sparse, dense, hybrid) can be selected or reconfigured without modifying the underlying codebase. *Re-ranking* indicates whether the system implements a dedicated re-ranking stage over the initial retrieval results. *Retrieval Evaluation* indicates whether the work provides a dedicated, retrieval-specific evaluation protocol, as opposed to reporting only end-to-end QA performance; the term *Partial* denotes approaches that report retrieval-related metrics within their experimental study but do not provide reusable tools or dedicated workflows for systematic retrieval benchmarking. *Reusable Framework* indicates whether the system is released as general-purpose, deployable software applicable beyond the original corpus and use case, rather than as a study-specific implementation.

Work	Configurable Retrieval Pipeline	Re-ranking	Retrieval Evaluation	Reusable Framework
KRA-GEN [21]	✗	✓	Partial	✗
BAILICAI [8]	✗	✓	Partial	✗
TC-RAG [17]	✗	✓	Partial	✗
MIRAGE [33]	✗	✗	✓	✓
Self-BioRAG [16]	✗	✓	Partial	✗
MedRAG [34]	✗	✓	Partial	✗
RAGtio (proposed framework)	✓	✓	✓	✓

for small or specialised corpora, where it may lead to suboptimal configuration choices. The evaluation framework presented here tackles the problem by offering two complementary and explicitly separated evaluation modes: Mode A, which automatically generates synthetic queries from randomly sampled chunks via an LLM to enable rapid large-scale assessment, and Mode B, which accepts user-provided query relevance pairs assembled from human expertise, yielding an unbiased estimate of retrieval quality on genuinely novel queries.

Frameworks, tooling and reproducibility. Despite the proliferation of RAG prototypes, the ecosystem remains highly fragmented, with heterogeneous techniques, metrics, and pipeline designs that hinder comparability and reproducibility across studies [5]. In the biomedical and healthcare domain specifically, the engineering complexity of RAG pipelines and the technical expertise required for deployment have been identified as primary barriers to clinical adoption [1], a limitation that extends to domain-specific applications more broadly [17, 8].

The absence of standardised, easily deployable software for evaluating retrieval components remains a recurring limitation highlighted in recent studies [5, 14]. Although frameworks such as Haystack [9] provide flexible abstractions for building retrieval and generation pipelines, they do not define dedicated protocols for systematic retrieval evaluation, nor do they provide integrated support for reproducible benchmarking through graphical interfaces or configurable evaluation workflows.

Building upon Haystack and Qdrant, the framework presented in this work addresses these limitations by integrating modular retrieval experimentation, retrieval-specific evaluation, and accessible deployment within a single system. In contrast to many existing domain-specific solutions, the proposed framework is designed to be domain-agnostic and easily configurable, supporting heterogeneous document formats, pluggable embedding models, multiple retrieval configurations, and reproducible evaluation pipelines through a web-based GUI, REST API, and YAML-based configuration.

3 System Architecture and Implementation

RAGtio implements a modular Retrieval-Augmented Generation (RAG) framework. The architecture is organised into five main stages: document ingestion, indexing, retrieval, answer generation, and evaluation, as depicted in Figure 1. The following subsections describe each stage both from a design and an implementation standpoint, detailing the Haystack components that realise them; additional low-level implementation details

are provided in the corresponding appendices.

RAGtio Reproducible retrieval evaluation for domain-specific RAG

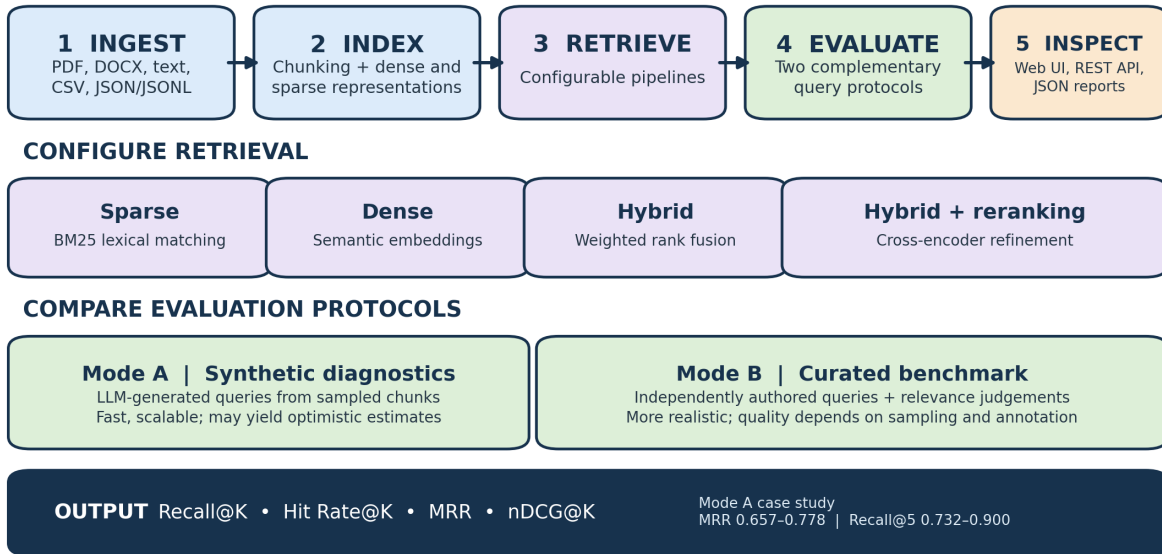


Figure 1: System architecture and evaluation workflow of the RAGtio framework. The modular end-to-end pipeline comprises five stages built on Haystack and Qdrant: document ingestion, indexing, retrieval, answer generation, and evaluation. The retrieval stage compares four configurations: sparse (BM25), dense (semantic embeddings), hybrid (weighted Reciprocal Rank Fusion), and hybrid with cross-encoder re-ranking. The evaluation stage supports two complementary modes: automated LLM-driven synthetic query generation (Mode A) and human-curated benchmarks (Mode B), assessed through four information retrieval metrics (Recall@K, Hit Rate@K, MRR, nDCG@K).

The system is designed as a set of loosely coupled pipelines that can be configured and combined independently, enabling flexible experimentation with different retrieval strategies and evaluation settings. Documents are first processed and indexed into a persistent Qdrant vector database. At query time, the selected retrieval configuration retrieves the most relevant document chunks, which are then provided to the LLM to generate the final answer. The entire workflow is orchestrated through the Haystack 2 framework and its modular pipeline abstraction [9]. RAGtio is deployed as a Docker Compose application integrating Qdrant, Ollama, and the main application services within an isolated internal network. This containerised design simplifies deployment, improves reproducibility, and enables consistent execution across different environments. The complete service specification is provided in the Supplementary Material, Section S1.1. The Docker Compose services, underlying technologies, and deployment roles are summarised in Supplementary Table S1, located immediately after the service-dependency description.

3.1 Document Ingestion

The ingestion stage transforms raw text files into Document Objects, each encapsulating:

- *Content*: the main text body used for both semantic and lexical retrieval.
- *Metadata*: auxiliary attributes such as identifiers, timestamps, source type, or domain-specific annotations.

The system supports heterogeneous input formats including plain text, Markdown, PDF, DOCX, CSV, and JSON/JSONL, enabling application to diverse real-world document collections without pre-processing.

Before storage, documents are processed by a **Document Cleaner** component, which removes empty lines, normalises whitespace, and filters boilerplate content, and are subsequently split into overlapping chunks by a configurable **Document Splitter (Chunker)** component, described in Section 3.2. Chunk size and overlap are exposed as YAML parameters, allowing adaptation to different document types without code changes [9].

3.2 Chunking Strategies

Chunking represents a critical design choice in RAG systems, as it defines the retrieval granularity and directly affects both retrieval precision and recall. The framework supports three configurable chunking strategies, selectable through a YAML parameter:

- **Character split:** splits text at a fixed character count with an optional overlap window. Useful as a controlled baseline in ablation studies, as it ensures a uniform chunk-size distribution.
- **Recursive split** (*default*): hierarchical splitting that preserves semantic units by attempting separators in priority order: paragraph boundary, line break, word boundary, and character level as a fallback.
- **Paragraph split:** one document paragraph becomes one chunk. Optimal for structured documents (academic papers, legal texts) where paragraphs are deliberately written as semantic units, but may degrade recall if very long paragraphs contain multiple distinct facts.

Additional implementation details are provided in the Supplementary Material, Section S2.1.

3.3 Indexing

Dense Embeddings Structured document chunks are converted into dense vector representations through sentence-transformer-based embedding models [25, 11]. The framework is model-agnostic and supports any HuggingFace compatible encoder [24], enabling flexible experimentation with both general-purpose and domain-specific embedding models while remaining suitable for CPU-based inference. At the implementation level, dense embeddings are generated through a *Dense Embedder* component built on FastEmbed ONNX-compatible sentence-transformer models [25, 24].

Model	Language	Dim.	Notes
intfloat/ multilingual-e5-large	Multilingual	1024	Default; instruction-tuned
intfloat/ multilingual-e5-base	Multilingual	768	Lighter alternative
BAAI/bge-m3	Multilingual	1024	State-of-the-art multi-function
BAAI/bge-small-en-v1.5	English	384	Efficient, English-only
sentence-transformers/ all-MiniLM-L6-v2	English	384	Fast, general purpose
pritamdeka/ S-PubMedBert-MS-MARCO	English (biomedical)	768	Fine-tuned on PubMed + MS MARCO

Table 2: Dense embedding models supported by the framework. *Model* reports the HuggingFace identifier of the encoder; *Language* indicates the language(s) the model was primarily trained and evaluated on; *Dim.* reports the dimensionality of the output embedding vector, which determines the vector size of the corresponding Qdrant collection; *Notes* summarises the intended use case or distinguishing characteristic of the model (e.g., default configuration, resource efficiency, domain specialisation). Additional custom embedding models can be integrated through the `embedder.custom` configuration block.

Table 2 summarises the six dense embedding models currently supported by the framework, spanning multilingual, lightweight, and biomedical-specific encoders. `intfloat/multilingual-e5-large` is used as the de-

fault, offering strong multilingual retrieval at the cost of higher latency, while `intfloat/multilingual-e5-base` and `BAAI/bge-small-en-v1.5` provide lighter alternatives for resource-constrained settings. `BAAI/bge-m3` additionally supports sparse and multi-vector retrieval, `sentence-transformers/all-MiniLM-L6-v2` serves as a fast general-purpose baseline, and `pritamdeka/S-PubMedBert-MS-MARCO` is fine-tuned on biomedical literature for domain-specific corpora. The embedding dimensionality determines the vector size of the corresponding Qdrant collection and is set automatically at indexing time based on the selected model. The complete implementation-level list of supported dense encoders is reported in Supplementary Table S3, located in the Supplementary Material, Section S2.2.1, after the description of dense embedding generation.

The generated embeddings, together with their associated metadata, are stored in a Qdrant collection using cosine similarity indexing. This separation between embedding generation and vector storage keeps the retrieval pipeline modular and allows alternative storage backends to be integrated without modifying the overall pipeline logic [9, 32].

Sparse Representations In parallel with dense embeddings, the system derives sparse representations using BM25 [26], computed by a dedicated *Sparse Embedder* component that produces BM25-compatible sparse lexical representations [26]. BM25 scoring is defined as:

$$\text{score}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \left(1 - b + b \cdot \frac{|D|}{\text{avgl}}\right)}, \quad (1)$$

where $f(q_i, D)$ is the frequency of term q_i in document D , $|D|$ is the document length, avgl is the average document length, $\text{IDF}(q_i)$ is the inverse document frequency, and k_1, b are hyperparameters controlling term-frequency saturation and length normalisation [26]. Sparse representations are particularly effective for handling rare terminology, acronyms, and specialised domain-specific tokens that may be inadequately captured by dense embedding models, especially in biomedical corpora.

Indexing Pipeline Document chunks, together with their dense and sparse representations, are written to Qdrant by a *Qdrant Writer* component using configurable duplicate-handling policies, supporting both incremental indexing and full collection rebuilding [32]. Figure 2 summarises the complete indexing data flow, from document cleaning to storage.

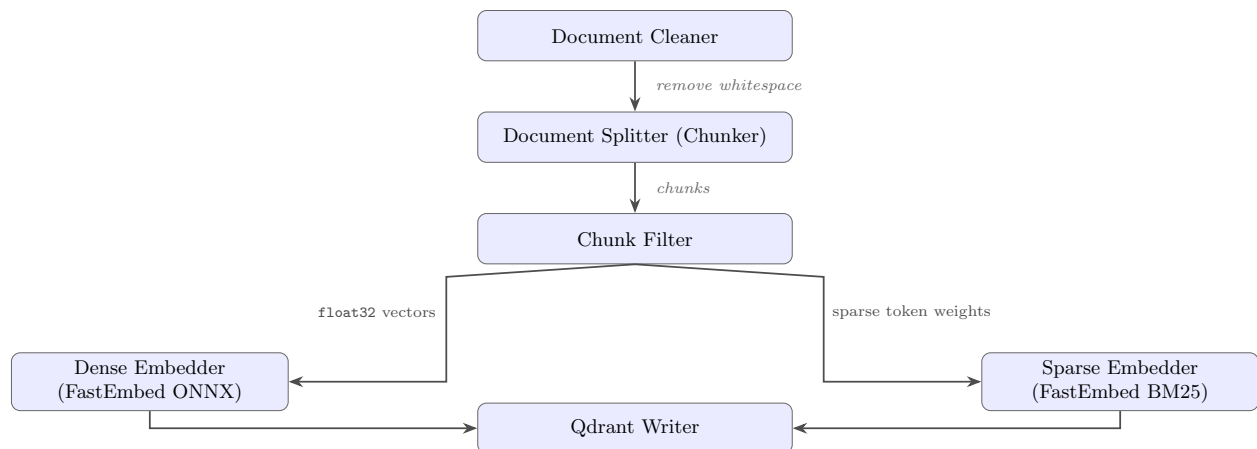


Figure 2: Indexing pipeline data flow.

To guarantee consistent indexing behaviour across repeated executions, chunk identifiers are deterministically generated from the source document path and the chunk position within the document, enabling stable deduplication and reproducible indexing. Additional implementation details are provided in the Supplementary Material, Section S2.

3.4 Retrieval

At query time, the input query is processed through the selected retrieval configuration over the same indexed document collection, enabling direct comparison across different retrieval paradigms. The framework implements four retrieval configurations, each realised as a distinct, independently instantiable pipeline built from shared, modular Haystack components and selectable via YAML configuration without code changes:

- **Sparse:** BM25-based lexical retrieval, implemented through Qdrant’s native BM25 retriever component, accepting either natural-language or LLM-generated keyword queries; the two query modes share the same retriever and differ only in the query pre-processing step. This configuration represents the standard lexical retrieval baseline.
- **Dense:** dense semantic retrieval based on sentence-transformer embeddings and cosine similarity search, implemented through a Haystack embedding retriever that wraps the HuggingFace-compatible encoder selected at indexing time (Section 3.3) and queries the same Qdrant collection.
- **Hybrid:** a combined dense-sparse retrieval strategy using weighted Reciprocal Rank Fusion (RRF), implemented by instantiating the sparse and dense retriever components in parallel within a single pipeline branch and joining their outputs through a dedicated rank-fusion component.
- **Hybrid with re-ranking:** the hybrid configuration extended with an additional cross-encoder re-ranking component, applied after rank fusion to the top- k fused candidates for further precision refinement.

All four configurations share the same document index, top- k parameter, and query set within each evaluation, ensuring a fully controlled comparison; only the internal composition of retriever and (optionally) re-ranker components differs between configurations.

In the hybrid configuration, the query is processed simultaneously through both the dense and sparse retrieval branches. The dense branch encodes the query using the same sentence transformer model adopted during indexing, ensuring consistency within the shared embedding space [25], while the sparse branch applies BM25-based lexical retrieval directly over the indexed corpus. The ranked outputs produced by the two branches are then combined through weighted Reciprocal Rank Fusion (RRF) [7], defined as:

$$\text{RRF}_w(d) = \sum_{r \in R} \frac{w_r}{k + \text{rank}_r(d)}, \quad w_{\text{dense}} = 0.7, \quad w_{\text{sparse}} = 0.3. \quad (2)$$

where $R = \{\text{dense}, \text{sparse}\}$ is the set of retrieval branches being fused, $\text{rank}_r(d)$ is the rank position of document d within the ranked list returned by branch r , w_r is the weight assigned to branch r , and k is a smoothing constant (default 60) that reduces the influence of top-ranked documents from either branch and stabilises the fusion for documents ranked lower in one of the two lists. The dense branch is weighted higher ($w_{\text{dense}} = 0.7$) to favour semantic relevance while preserving the lexical robustness provided by sparse matching ($w_{\text{sparse}} = 0.3$); these default weights are exposed as configurable parameters, so alternative ratios can be evaluated without code changes.

The implementation-level hybrid retrieval pipeline is illustrated in Supplementary Figure S1, located in the Supplementary Material, Section S3.1.3, after the description of the hybrid retrieval mode. An optional cross-encoder re-ranker can be applied after the rank fusion for additional precision refinement, using a two-stage retrieve-then-rerank paradigm. The two-stage workflow is illustrated in Supplementary Figure S2, followed by the architectural comparison in Supplementary Table S4, both located in Section S3.2 after the retrieve-then-rerank description. Additional implementation details, including component-level configuration parameters and default values, are provided in the Supplementary Material, Section S3.

3.5 Query Enhancement

Single-query retrieval suffers from the *vocabulary mismatch problem*: the user’s phrasing may not match document terms even when semantically equivalent. RAGtio addresses this issue through two optional query enhancement techniques that can be applied before retrieval:

- **Query expansion:** an LLM generates n semantically equivalent paraphrases of the original query. Retrieval is performed for all variants and results are merged by deduplication on chunk ID. This strategy improves recall by increasing the lexical and semantic coverage of the search process.
- **Query decomposition:** complex multi-hop questions are split into n independent sub-questions, each retrievable separately. This approach is particularly useful when relevant evidence is distributed across multiple non-overlapping document segments [4].

Both techniques are disabled by default and can be enabled independently via YAML configuration. When both are active, decomposition is applied first, and expansion is applied only to the original query (not to sub-queries) to avoid a combinatorial explosion of LLM calls.

3.6 Answer Generation

Once a retrieval configuration has been selected, the generation stage retrieves the top- k relevant document chunks and constructs the prompt provided to the LLM. Retrieved passages are organised into a numbered context block and prepended to the original user question, enabling grounded answer generation based on the retrieved evidence.

The retrieved evidence is drawn exclusively from the document collection previously indexed during the ingestion stage. As the framework is domain-agnostic, this collection may consist of scientific publications on diabetes, cancer, drugs, and infectious diseases, or any other corpus processed through the ingestion pipeline.

The system adopts a two-message chat format:

- a *system message*, defining the assistant’s role and grounding instructions;
- a *user message*, containing the retrieved context block and the original question.

The system prompt explicitly instructs the model to remain faithful to the provided context and to acknowledge when the available information is insufficient, helping mitigate hallucination effects [18]. An additional language-adaptive instruction (“answer in the same language as the question”) enables multilingual operation without requiring an explicit language-detection stage.

The framework supports two alternative LLM backends, selected via configuration through a unified interface without modifying the remaining pipeline:

- **Ollama** (local inference), enabling fully self-hosted deployment for sensitive enterprise, biomedical, or clinical documents where external cloud APIs may be restricted;
- **OpenAI-compatible APIs** (cloud inference), supporting OpenAI and compatible endpoints through configurable API settings.

Both backends support streaming responses through Server-Sent Events (SSE), allowing tokens to be returned incrementally and improving interaction responsiveness. At the interface level, the web application exposes an interactive chat that submits user questions to the selected retrieval pipeline, assembles the retrieval context, and streams the generated answer to the client through the same SSE mechanism. The complete RAG Q&A pipeline is illustrated in Supplementary Figure S3, located in the Supplementary Material, Section S4.1, immediately after the introduction of the pipeline stages. Additional implementation details, including the complete prompt templates and the **RAGResponse** schema exchanged between backend and frontend, are provided in the Supplementary Material, Section S4.

3.7 Evaluation

The evaluation stage assesses retrieval quality across the four retrieval configurations, using standard Information Retrieval (IR) metrics, and introduces two complementary evaluation modes, designed respectively for rapid automated assessment and unbiased, expert-curated benchmarking.

Information Retrieval Metrics Retrieval quality is assessed using four standard IR metrics, all implemented from scratch without external evaluation libraries to ensure full reproducibility and transparency:

- **Recall@K**: proportion of relevant documents found in the top- K retrieved results.
- **Hit Rate@K**: proportion of queries for which at least one relevant document appears within the top- K retrieved results. In Mode A, where $|\text{relevant}| = 1$, Hit Rate@ K becomes equivalent to Recall@ K .
- **Mean Reciprocal Rank (MRR)**: average of the reciprocal rank of the first relevant document. MRR is most appropriate for navigational queries where users inspect results in order and stop at the first satisfactory item.
- **nDCG@K** (Normalised Discounted Cumulative Gain): measures ranking quality by weighting relevant documents according to their position. nDCG@ K is the primary metric used for automatic retriever selection.

A complete overview of the metric formulations is provided in the Supplementary Material, Section S6. Practical guidance for metric selection is summarised in Supplementary Table S7, located in Section S6.5 after the individual metric definitions.

Mode A: Synthetic Query Generation In Mode A, an LLM generates synthetic evaluation queries from a randomly sampled subset of the indexed document chunks. For each document, two queries are produced: a natural-language query (used to evaluate `sparse_nl`, `dense`, and `hybrid` configurations) and a keyword query (used to evaluate `sparse_kw`). This distinction reflects the retrieval characteristics of the evaluated retrievers: sparse keyword retrieval benefits from explicit lexical overlap, whereas dense and hybrid retrievers are better assessed through natural-language questions that exercise semantic generalisation capabilities.

Each generated query is paired with the source chunk that created it, yielding an automatically constructed benchmark for retrieval evaluation [33]. To improve efficiency and reproducibility, generated query sets are cached on disk and reused across subsequent evaluation runs on the same corpus. The available query-generation modes are summarised in Supplementary Table S5, located in the Supplementary Material, Section S5.1, after the Mode A evaluation algorithm. The retrieval-aware strategy associated with each query-type setting is reported in Supplementary Table S6, located in Section S5.1.1 after the adaptive query-resolution procedure and prompt templates.

Mode A requires no external annotation and is fully automated, making it the default configuration for rapid pipeline assessment. However, this evaluation mode is inherently affected by *self-referential bias*, since the generated queries originate directly from the indexed corpus itself. As a result, the retrieval task may become artificially easier than in real-world usage scenarios.

Mode B: User-Provided Query-Relevance Pairs Mode B addresses the self-referential bias by accepting a user-provided labelled dataset of query-relevance pairs. The input is a JSON array where each entry specifies a query, a list of relevant chunk identifiers, and an optional reference answer:

```
[
  {
    "query": "What is the recommended dosage for compound X?",
    "relevant_ids": ["chunk-abc-123", "chunk-def-456"],
    "expected_text": "optional reference answer"
  },
  ...
]
```

For each query, retrieval is executed using the same evaluation metrics adopted in Mode A, but without relying on LLM-generated queries. Because the queries are assembled from domain expertise and the relevant chunk identifiers are manually curated, Mode B provides an unbiased estimate of retrieval quality on genuinely novel queries that are not guaranteed to mirror the phrasing of indexed documents.

The coexistence of both evaluation modes within the same framework enables direct comparison between synthetic and expert-curated benchmarks, quantifying the impact of self-referential bias on retrieval performance. For specialised biomedical corpora, where this bias is particularly pronounced, Mode B represents the most reliable evaluation setting. Additional implementation details are provided in the Supplementary Material, Section S5.

4 Case Study

4.1 Experimental Design

The experimental evaluation is organised around four biomedical domains, with five papers per domain manually collected from open-access biomedical literature to provide heterogeneous terminology, document structure, and retrieval complexity. For each domain, the five indexed documents are evaluated along three complementary axes, summarised in Table 3: a small set of interactive question-answer examples used to qualitatively illustrate end-to-end answer generation (Section 3.6), a large set of automatically generated queries used for large-scale retrieval evaluation under Mode A, and a small set of expert-curated queries used for unbiased retrieval evaluation under Mode B. This combination allows the same indexed corpus to support both a qualitative demonstration of the full RAG pipeline and a quantitative, statistically meaningful comparison of the four retrieval configurations.

Table 3: Composition of the evaluation corpus across the four biomedical domains. *#Docs* reports the number of source documents indexed per domain. *#QA Examples* reports the number of interactive question-answer examples used to qualitatively illustrate end-to-end answer generation (Section 3.6), rather than retrieval evaluation. *#Mode A Queries* reports the number of LLM-generated synthetic queries used for large-scale retrieval evaluation (Section 3.7). *#Mode B Queries* reports the number of manually curated, expert-style queries with associated relevant chunk identifiers used for unbiased retrieval evaluation.

Domain	#Docs	#QA Examples	#Mode A Queries	#Mode B Queries
Oncology	5	5	41	5
Diabetes	5	5	40	5
Pharmacology	5	5	40	5
Infectious Diseases	5	5	38	5
Total	20	20	159	20

For each domain, the five indexed documents are evaluated using both retrieval evaluation modes introduced in Section 3.7:

- **Mode A (synthetic evaluation):** automatically generated queries are used to assess large-scale retrieval behaviour and compare retrieval configurations under controlled conditions.
- **Mode B (manual evaluation):** five manually curated expert-style queries are defined for each domain together with their relevant chunk identifiers, enabling a more realistic evaluation setting and reducing self-referential bias.

All experiments within each domain are executed over the same indexed corpus and under identical retrieval settings to ensure fair and controlled comparison.

The indexing configuration remains constant across all experiments:

- **Chunking strategy:** recursive;
- **Embedding model:** `intfloat/multilingual-e5-large` (FastEmbed);
- **Vector database:** Qdrant with cosine similarity indexing.

During retrieval, the system operates in *Hybrid mode* (dense + sparse), with default parameters and re-ranking enabled. Retrieval quality is evaluated using Recall@K, Hit Rate@K, MRR, and nDCG@K, with

nDCG@K adopted as the primary ranking metric for comparative analysis.

Unless otherwise specified, all experiments were conducted using the hybrid retrieval pipeline. This choice was motivated by the current state of the literature, where hybrid retrieval is generally regarded as a strong baseline because it combines the semantic matching capabilities of dense retrieval with the exact lexical matching provided by sparse retrieval.

Since the objective of the case study is to evaluate the proposed framework as an end-to-end system rather than to exhaustively benchmark retrieval paradigms, the experimental analysis focuses on the hybrid configuration, which combines lexical and semantic signals and is consistently reported in the literature as the most robust default across heterogeneous corpora and query types [20, 15, 10]. The alternative configurations available as independently selectable pipelines are part of the framework’s contribution, allowing users to empirically identify the best-performing configuration for their own corpus.

4.2 Quantitative Results

Table 4 and Table 5 summarise the retrieval performance obtained with the hybrid retrieval configuration combined with cross-encoder re-ranking across the four biomedical domains. Overall, the results show consistent effectiveness across Oncology, Diabetes, Pharmacology, and Infectious Diseases, although moderate variability is observed depending on corpus characteristics and query complexity.

Table 4: Mode A (synthetic evaluation) aggregate results across four biomedical domains (hybrid retrieval with cross-encoder re-ranking). Perfect/Partial/Failed refer to the number of queries with $MRR=1.0$, $0 < MRR < 1$, and $MRR=0$, respectively.

Domain	#Q	MRR	Recall@5	Recall@10	Perfect	Partial	Failed
Oncology	41	0.657	0.732	0.732	25	5	11
Diabetes	40	0.661	0.875	0.875	20	15	5
Pharmacology	40	0.778	0.900	0.900	28	8	4
Infectious Diseases	38	0.751	0.895	0.895	24	10	4

Table 5: Mode B (manual evaluation) aggregate results across four biomedical domains (hybrid retrieval with cross-encoder re-ranking).

Domain	#Q	MRR	R@1	R@5	R@10	nDCG@5
Oncology	5	0.800	0.132	0.381	0.381	0.533
Diabetes	5	0.700	0.250	0.450	0.450	0.489
Pharmacology	5	0.540	0.073	0.427	0.427	0.389
Infectious Diseases	5	0.650	0.122	0.273	0.273	0.390

The Oncology domain is used as the primary case study in the discussion below, while the complete per-query results for all domains are reported in the Supplementary Material, Section S7.

Mode A - Synthetic evaluation. As shown in Table 6, the hybrid retrieval configuration achieves solid performance on the Oncology corpus. The ranking metrics indicate that relevant chunks are generally surfaced early in the result list, reflecting both effective dense retrieval and the contribution of cross-encoder re-ranking in promoting semantically aligned candidates.

The overall ranking quality remains stable across cutoffs, suggesting that the system is able to prioritise the correct chunk even when multiple semantically similar passages are present in the corpus. Moreover, the overlap between Recall@5 and Recall@10 indicates that retrieving more than five candidates does not yield further gains in this setting.

Table 6: Mode A (synthetic evaluation) results for the Oncology domain using the hybrid retrieval with cross-encoder re-ranking configuration.

Metric	@1	@3	@5	@10
MRR	0.657	—	—	—
Recall	0.610	0.683	0.732	0.732
Hit Rate	0.610	0.683	0.732	0.732
nDCG	0.610	0.656	0.676	0.676

Mode B - Manual evaluation. In Mode B, the expert-based evaluation reveals a mixed picture. As shown in Table 7, the system is highly effective at surfacing at least one relevant passage immediately, confirming that the re-ranking step captures the core intent of the query.

Table 7: Mode B (manual evaluation) results for the Oncology domain using the hybrid retrieval with cross-encoder re-ranking configuration (5 manually curated queries).

Metric	@1	@3	@5	@10
MRR	0.800	—	—	—
Recall	0.132	0.249	0.381	0.381
Hit Rate	0.800	0.800	0.800	0.800
nDCG	0.800	0.528	0.533	0.454

However, when considering how many of the expert-annotated passages appear within the top retrieved results, performance naturally declines. This is largely due to the structure of Mode B itself: each query is associated with a broad set of relevant chunks, far more numerous than in Mode A. As a result, even consistently retrieving the best-matching chunk at rank 1 cannot ensure full coverage within a limited retrieval window, making lower recall and nDCG values an expected outcome rather than a system deficiency.

Interactive Q&A. The interactive Q&A component was evaluated using five representative questions spanning different aspects of oncology, including tumour biology, immunotherapy, and precision oncology. Table 8 summarises the retrieval confidence of the supporting passages together with the qualitative assessment of the generated answers.

Table 8: Interactive Q&A sessions for the Oncology domain. Hybrid retrieval with cross-encoder re-ranking (Top- $K=10$, Top- $N=5$ after rerank). Reranker scores are reported for each retrieved chunk; the mean score summarises overall retrieval confidence. Answer quality is assessed qualitatively by a domain expert (**C** = Correct, **P** = Partial, **F** = Failed).

Question	#Sources	s_1	s_2	s_3	s_4	$\bar{s}$	Quality
Q1	3	0.999	0.420	0.378	—	0.599	C
Q2	4	1.000	0.500	0.450	0.375	0.581	P
Q3	4	0.992	0.880	0.854	0.686	0.853	C
Q4	3	1.000	0.999	0.998	—	0.999	C
Q5	4	0.995	0.995	0.992	0.985	0.992	C

s_i : reranker score of the i -th retrieved chunk; $\bar{s}$: mean score across all retrieved chunks.

C = Correct, **P** = Partial (retrieved chunks lack definitional content).

Overall, the interactive Q&A component produced four fully correct answers and one partially correct answer. Queries associated with high reranker confidence, particularly Q4 and Q5, consistently resulted in accurate responses supported by highly relevant evidence. The only partially correct case (Q2) reflects a limitation of the retrieved passages rather than a retrieval failure, as the returned evidence addressed the topic but did

not provide a complete definition of antibody-drug conjugates.

The evaluation includes the following five representative questions:

- Q1.** *What is the central problem for tumour immunology research?*
- Q2.** *What are antibody-drug conjugates?*
- Q3.** *What is the structure of a CAR-T cell?*
- Q4.** *How do tumour cells contribute to resistance to immunotherapy?*
- Q5.** *Are there robust predictive and prognostic biomarkers for breast cancer?*

Together, these questions span both broad conceptual issues and more specific molecular and clinical topics, providing a diverse evaluation of the framework under different information needs.

All qualitative judgments reported in Table 8 were assigned by a single domain expert using a three-level rubric: *Correct* (factually accurate and fully supported by the retrieved passages), *Partially Correct* (containing correct information but incomplete or only partially grounded in the retrieved evidence), and *Failed* (incorrect, unsupported, or contradicted by the retrieved documents). The use of a single annotator represents a limitation of the present study, and future work will incorporate multiple domain experts to evaluate answer quality and measure inter-rater agreement.

Figures 3 and 4 illustrate the interactive Q&A interface using Query 5 as a representative example. The first figure shows the dashboard presented to the user, whereas the second displays the generated answer together with the retrieved supporting passages used to ground the response.

5 Discussion

RAGtio’s central contribution is its integration of automated evaluation, manual assessment, and interactive Q&A within a single, user-friendly platform. While systems such as MIRAGE [33] provide comprehensive retrieval benchmarking and domain-specific frameworks including KRA-GEN [21], TC-RAG [17], BAILICAI [8], Self-BioRAG [16], and MedRAG [34] focus on specific biomedical applications, none combines configurable retrieval strategies, retrieval-specific evaluation, and interactive testing within a unified environment. RAGtio addresses this gap through a web-based interface that enables domain experts, such as clinicians and biomedical researchers, to evaluate RAG performance on their own corpora without writing code, annotate relevant passages for Mode B evaluation through a visual interface, and interactively test the system via natural-language questions with immediate retrieval and generation feedback. This combination of rigorous evaluation methodology and accessibility distinguishes the proposed framework from existing biomedical RAG platforms.

The framework’s automatic retriever selection, which identifies and uses the configuration with the highest $nDCG@K$ for the interactive Q&A step, embeds a testable hypothesis: retrieval metrics predict final answer quality. In the annotated sample considered in this study (20 queries across four biomedical domains), a qualitative association between high mean reranker scores and correct answers was observed: queries with $\bar{s} > 0.9$ generally produced *Correct* answers, whereas substantially lower reranker scores were associated with weaker retrieval evidence and reduced answer quality. However, this relationship is not deterministic: the LLM can occasionally compensate for moderately noisy retrieval when the highest-ranked passage is highly relevant (e.g., Oncology Q1, where $\bar{s} = 0.599$ but the top retrieved passage achieved $s_1 = 0.999$). These observations support the use of retrieval metrics as practical indicators of downstream answer quality, while also motivating future work on integrating generation-quality metrics into the retriever selection process.

The framework’s modular architecture allows independent modification of every pipeline component without affecting the remaining stages. Switching the embedding model, for example, requires only a single YAML configuration change, while alternative rank-fusion strategies, additional information retrieval metrics, or new evaluation modes can be incorporated through standardised module interfaces. Unlike existing biomedical RAG systems, whose pipelines are typically tailored to a specific corpus or downstream task [17, 8, 16, 34], RAGtio enables controlled comparisons of sparse, dense, hybrid, and re-ranked retrieval configurations

Question

Are there robust predictive and prognostic biomarkers for breast cancer?

RETRIEVAL OPTIONS
Mode

☐ Dense
☐ Sparse
☒ Hybrid

Top-K

10

Top-N (after rerank)

5

☒ Reranker enabled

METADATA FILTERS

Field
=
Value
+

QUERY ENHANCEMENT

☒ Query Expansion

3

variants

☒ Query Decomposition

3

sub-queries

Send Question

☒ Streaming SSE

Figure 3: Dashboard for Query 5 (*Are there robust predictive and prognostic biomarkers for breast cancer?*) in the interactive Q&A mode.

within the same experimental environment. This flexibility facilitates reproducible longitudinal studies in which retrieval components can be progressively upgraded while maintaining directly comparable evaluation conditions.

The emphasis on reproducibility further distinguishes the proposed framework. All stochastic processes are controlled through a fixed random seed, while generated queries, retrieval results, evaluation metrics, and retriever-selection outcomes are serialised as structured JSON artefacts to support offline analysis and direct comparison across experimental runs. This complements benchmarking-oriented frameworks such as MIRAGE [33], extending them with configurable retrieval pipelines, interactive evaluation capabilities, and an integrated web-based interface suitable for deployment by users without extensive programming expertise. The complete framework, including reproducible source code and the web application, is publicly available at <https://github.com/ArchAI-Labs/ragtio>.

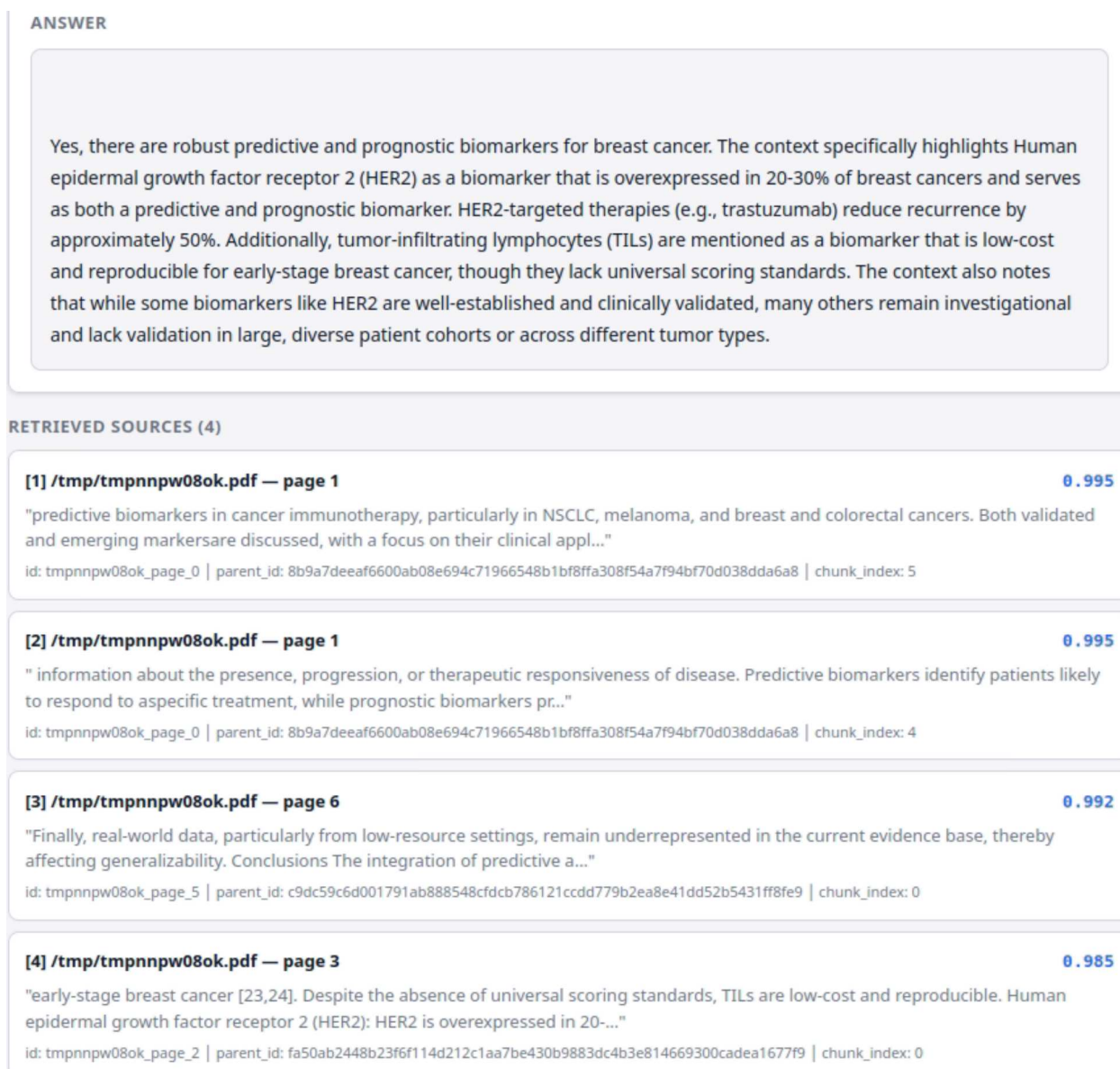


Figure 4: Generated answer and retrieved supporting passages for Query 5 in the interactive Q&A mode.

6 Limitations

Although the proposed framework is designed to be general and extensible, the present study has some limitations that should be considered when interpreting the results.

First, the manual evaluation (Mode B) was conducted on a relatively small number of annotated queries. While this choice reflects the substantial effort required for expert annotation, it limits the statistical strength of the observed trends.

Second, the case study evaluates a single embedding model. The framework itself is embedding-agnostic and supports alternative dense encoders through configuration only; however, the current experimental validation does not compare different embedding families. Consequently, the reported retrieval performance should not be interpreted as representative of all available embedding models.

Third, although the framework supports dense-only, sparse-only, and hybrid retrieval pipelines, the present

case study focuses exclusively on the hybrid configuration. This choice was motivated by the fact that hybrid retrieval combines the complementary strengths of dense semantic matching and sparse lexical matching, and is therefore commonly regarded as a competitive retrieval strategy. Furthermore, while the framework enables systematic comparison of multiple retrieval configurations, the interactive case study evaluates only the configuration automatically selected as the best according to retrieval metrics. Consequently, a comprehensive ablation study involving different retrieval paradigms, embedding models, chunking strategies, and reranking configurations was outside the scope of the present work and remains an important direction for future research.

Finally, the manual relevance annotations were produced by a single human annotator. While this ensured consistency throughout the evaluation process, it does not allow assessment of inter-annotator agreement or annotation variability. Future studies should involve multiple domain experts and quantify annotation reliability using established agreement measures.

7 Conclusions

This paper has presented RAGtio, a modular and open-source framework for the systematic and reproducible evaluation of RAG-based retrieval pipelines, with primary application to the biomedical domain. The framework integrates three complementary functionalities within a single web-based platform: automated synthetic evaluation (Mode A), expert-curated manual evaluation (Mode B), and interactive Q&A with live retrieval and generation.

Evaluation across four biomedical domains shows that the framework consistently produces meaningful and actionable diagnostics. The hybrid retrieval pipeline, supported by cross-encoder re-ranking, exhibits stable ranking behaviour.

The dual evaluation protocol highlights a domain-dependent self-referential bias: synthetic queries tend to overestimate retrieval quality in some corpora, whereas manually-annotated queries reveal stronger top-rank accuracy in others. This confirms the importance of applying both evaluation modes within a unified framework.

Failure analysis further identifies recurring issues linked to metadata-only chunks, table-derived fragments, and broad thematic questions. These patterns point to chunking strategy as the main bottleneck and offer practical guidance for corpus preparation.

Finally, initial evidence from the interactive Q&A sessions supports the automatic retriever-selection mechanism, with higher re-ranker confidence aligning with more reliable answers.

The system is deployed as a Docker Compose application with a FastAPI REST backend, a Qdrant vector store, local LLM inference via Ollama, and a web-based GUI that lowers the barrier to adoption for non-technical users.

Future work will extend the comparison to larger manually annotated benchmarks involving multiple expert annotators, evaluate a broader range of embedding models and retrieval configurations through systematic ablation studies, investigate adaptive weighting schemes for dense and sparse retrievers based on query type, and explore cost-aware configurations that dynamically select retrieval strategies and re-ranking depth according to latency and resource constraints [7, 33, 29].

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Consent to Publish Declaration

Not applicable.

Consent to Participate Declaration

Not applicable.

Ethics Declaration

Not applicable.

Author Contributions

AF and PHG conceived the main idea of the study. NP led the software development. PHG and PVe led the project. PViz guided the software development and contributed to the system design. All authors contributed to the preparation of the manuscript, read the final version, and approved it for submission.

Availability of data and materials

The datasets used in this study were obtained from publicly available sources reported in the scientific literature. The corresponding publications were identified through Google Scholar, and the datasets were downloaded as described in the Methods section. No new datasets were generated during this study.

Data and Software Availability

The source code, configuration files, and materials supporting this study are publicly available in the RAGtio GitHub repository at <https://github.com/ArchAI-Labs/ragtio>.

Availability and Requirements

Project name: RAG Document Retrieval and Question Answering System

Project home page: <https://github.com/ArchAI-Labs/ragtio>

Operating system(s): Platform independent (Docker-based; Windows, Linux, macOS)

Programming language: Python

Other requirements:

Docker 24 or higher

Docker Compose 2.20 or higher (or Docker Desktop)

Alternatively: Rancher

Ollama (default model: qwen3:4b) or OpenAI API (optional)

License: Apache License 2.0

Any restrictions to use by non-academics: None (Apache-2.0 license permits commercial and non-academic use).

Supplementary Material

A separate supplementary file accompanies this manuscript. It contains detailed system architecture and implementation documentation, together with the complete per-query case-study results for all four biomedical domains.

References

- [1] Mohamed Abo El-Enen, Sally Saad, and Taymoor Nazmy. A survey on retrieval-augmentation (rag) models for healthcare applications. *Neural Computing and Applications*, 37(33):28191–28267, 2025.
- [2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [3] Lameck Mbangula Amugongo, Pietro Mascheroni, Steven Brooks, Stefan Doering, and Jan Seidel. Retrieval augmented generation for large language models in healthcare: A systematic review. *PLOS Digital Health*, 4(6):e0000877, 2025.
- [4] Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199, 2024.
- [5] Andrew Brown, Muhammad Roman, and Barry Devereux. A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges. *arXiv preprint arXiv:2508.06401*, 2025.
- [6] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.
- [7] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759, 2009.
- [8] Long Cui, Yongbin Liu, Chunping Ouyang, Ying Yu, Jiangtao Zhang, Yaping Wan, and Fei Yang. Bailicai: A domain-optimized retrieval-augmented generation framework for medical applications. *Big Data Mining and Analytics*, 9(2):376–392, 2026.
- [9] deepset. Haystack: An open source nlp framework. <https://haystack.deepset.ai/overview/intro>, 2024. Accessed: 2024-11-27.
- [10] Annamaria Defilippo, Giovanni Canino, Nicola Procopio, Albino Trapuzzano, Sabato Sorrentino, Ciro Indolfi, Patrizia Vizza, Pierangelo Veltri, and Pietro Hiram Guzzi. Design of a rag framework for cardiology ehr analysis. *Discover Artificial Intelligence*, 2026.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186, 2019.
- [12] Pinar Ersoy and Mustafa Erşahin. A comparative evaluation of rag architectures for cross-domain llm applications: Design, implementation, and assessment. *IEEE Access*, 13:194185–194196, 2025.
- [13] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- [14] Md Toufique Hasan, Muhammad Waseem, Kai-Kristian Kemell, Ayman Asad Khan, Mika Saari, and Pekka Abrahamsson. Engineering rag systems for real-world applications: Design, development, and evaluation. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 143–158. Springer, 2025.
- [15] Hsin-Ling Hsu and Jengnan Tzeng. Dat: Dynamic alpha tuning for hybrid retrieval in retrieval-augmented generation. *arXiv preprint arXiv:2503.23013*, 2025.

- [16] Minbyul Jeong, Jiwoong Sohn, Mujeen Sung, and Jaewoo Kang. Improving medical reasoning through retrieval and self-reflection with retrieval-augmented large language models. *Bioinformatics*, 40(Supplement1):i119–i129, 07 2024.
- [17] Xinke Jiang, Yue Fang, Rihong Qiu, Haoyu Zhang, Yongxin Xu, Hao Chen, Wentao Zhang, Ruizhe Zhang, Yuchen Fang, Xinyu Ma, et al. Tc-rag: Turing-complete rag’s case study on medical llm systems. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11400–11426, 2025.
- [18] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
- [19] Siru Liu, Allison B McCoy, and Adam Wright. Improving large language model applications in biomedicine with retrieval-augmented generation: a systematic review, meta-analysis, and clinical development guidelines. *Journal of the American Medical Informatics Association*, 32(4):605–615, 2025.
- [20] Chandana Sree Mala, Gizem Gezici, and Fosca Giannotti. Hybrid retrieval for hallucination mitigation in large language models: A comparative analysis. *arXiv preprint arXiv:2504.05324*, 2025.
- [21] Nicholas Matsumoto, Jay Moran, Hyunjun Choi, Miguel E Hernandez, Mythreye Venkatesan, Paul Wang, and Jason H Moore. Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. *Bioinformatics*, 40(6):btac353, 2024.
- [22] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.
- [23] Gabrijela Perković, Antun Drobňjak, and Ilica Botički. Hallucinations in llms: Understanding and addressing challenges. In *2024 47th MIPRO ICT and electronics convention (MIPRO)*, pages 2084–2088. IEEE, 2024.
- [24] Nicola Procopio. Sentence-bert base italian xxl uncased. <https://huggingface.co/nickprock/sentence-bert-base-italian-xxl-uncased>, 2024. Accessed: 2024-11-27.
- [25] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.
- [26] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
- [27] Chaitanya Sharma. Retrieval-augmented generation: A comprehensive survey of architectures, enhancements, and robustness frontiers. *arXiv preprint arXiv:2506.00054*, 2025.
- [28] Aditi Singh, Abul Ehtesham, Saket Kumar, Tala Talaei Khoei, and Athanasios V Vasilakos. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.
- [29] Lukas Tuggener, Pascal Sager, Yassine Taoudi-Benchekroun, Benjamin F Grewe, and Thilo Stadelmann. So you want your private llm at home? a survey and benchmark of methods for efficient gpts. In *2024 11th IEEE Swiss Conference on Data Science (SDS)*, pages 205–212. IEEE, 2024.
- [30] Ozan Unlu, Jiyeon Shin, Charlotte J. Mailly, Michael F. Oates, Michela R. Tucci, Matthew Varugheese, Kavishwar Waghlikar, Fei Wang, Benjamin M. Scirica, Alexander J. Blood, et al. Retrieval augmented generation enabled generative pre-trained transformer 4 (gpt-4) performance for clinical trial screening. *medRxiv*, 2024.
- [31] Junde Wu, Jiayuan Zhu, Yunli Qi, Jingkun Chen, Min Xu, Filippo Menolascina, and Vicente Grau. Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. *arXiv preprint arXiv:2408.04187*, 2024.

- [32] Xingrui Xie, Han Liu, Wenzhe Hou, and Hongbin Huang. A brief survey of vector databases. In *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pages 364–371. IEEE, 2023.
- [33] Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251, 2024.
- [34] Xuejiao Zhao, Siyan Liu, Su-Yin Yang, and Chunyan Miao. Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 4442–4457, New York, NY, USA, 2025. Association for Computing Machinery.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [ragtiosupplementarymaterial.pdf](#)