A Hybrid RAG for Answering Questions about Technical Documentation from Github Projects

Gonzalo Librandi[1], Nicolás Miccio Palermo[2], Rafael Capilla[3], Antonela Tommasel[2], and Jorge Andrés Diaz-Pace[2]

[1] Facultad de Ciencias Exactas, UNCPBA University, Tandil, Buenos Aires, Argentina glibrandi@alumnos.exa.unicen.edu.ar
[2] ISISTAN Research Institute, UNCPBA University–CONICET, Tandil, Buenos Aires, Argentina
{nicolas.miccio, antonela.tommasel, andres.diazpace}@isistan.unicen.edu.ar
[3] Rey Juan Carlos University, Madrid, Spain
rafael.capilla@urjc.es

Abstract. Software professionals often have to answer technical questions about a development project, spanning design decisions and implementation artifacts and best practices, among others. Nonetheless, this information tends to be fragmented across PDFs, code repositories, and Web documentation. The widespread adoption of large language models (LLMs) to provide answers using Retrieval-Augmented Generation (RAG) techniques can support this challenge. However, existing solutions offer limited support for multi-source knowledge and provide weak traceability of sources for the generated content. In this context, we present a hybrid RAG system for software technical documentation that enables semantic search across heterogeneous sources, while preserving reference sources and reducing hallucinations. In this work, we developed a prototype of the hybrid RAG that supports multiple knowledge sources and evaluated it using a set of design-focused questions. An initial evaluation shows that our approach can provide accurate responses and a good source attribution, particularly in multi-source scenarios.

Keywords: knowledge management· LLM· software design

Un RAG Híbrido para Consultas sobre Documentación Técnica en Proyectos Github

Abstract. Los profesionales del software a menudo deben responder preguntas técnicas sobre un proyecto de desarrollo, que abarcan decisiones de diseño, artefactos de implementación y mejores prácticas, entre otros. Sin embargo, esta información suele estar fragmentada en

archivos PDF, repositorios de código y documentación web. La adopción de grandes modelos de lenguaje (LLMs) para proporcionar respuestas mediante técnicas de Generación Aumentada por Recuperación (RAG) puede ayudar a superar este desafío. No obstante, las soluciones existentes ofrecen un soporte limitado para múltiples fuentes de conocimiento y proporcionan una trazabilidad deficiente de las fuentes del contenido generado. En este contexto, se presenta un sistema RAG híbrido para documentación técnica de software que permite la búsqueda semántica en fuentes heterogéneas, preservando las fuentes de referencia y reduciendo las alucinaciones. En particular, se desarrolló un prototipo del sistema RAG híbrido que admite múltiples fuentes de conocimiento y se evaluó mediante un conjunto de preguntas relacionadas a diseño. Los resultados iniciales muestran que el enfoque proporciona respuestas precisas y una buena atribución de fuentes múltiples.

Keywords: gestión de conocimiento· LLM· diseño de software

1 Introduction

The reliance on technical documentation is a common practice in engineering and computer science domains. Practitioners and students regularly consult PDF textbooks, open-source repositories, API documentation, and specialized web resources (e.g., wikis, blogs, Web sites) simultaneously. However, technical documentation is often fragmented across multiple formats and platforms (Forward & Lethbridge, 2002). This fragmentation introduces a cognitive overhead, as users must navigate and analyze information from heterogeneous sources and manage high information diversity (Robillard et al., 2017). Today, an important aspect is the validation of a response in the sense that it is deemed correct and traceable to reputable sources. This activity often requires considerable human effort when performed manually (Daka & Fraser, 2014). For instance, if users look for a particular design pattern in a Github repository, they might want to know details of the generic pattern as well as the way it was actually implemented in the project.

Traditional approaches, such as keyword-based search mechanisms rely on lexical matching and fail to bridge the semantic gap when terminology differs between user queries and documentation. Nowadays, Large Language Models (LLMs) constitute an alternative mechanism to provide a conversational access to documentation. Nevertheless, LLMs are not without drawbacks due to their susceptibility to hallucinations and domain-specific inaccuracies, producing plausible but unsupported statements (Lewis et al., 2020). Furthermore, the lack of traceability in standalone LLMs prevents users from verifying claims against original sources, which is a critical requirement in professional engineering and educational contexts.

The Retrieval-Augmented Generation (RAG) approach addresses several of these limitations by integrating the LLM output with externally retrieved documentation (Gao et al., 2023). However, many RAG systems still focus on homogeneous corpora such as PDF collections or Web pages, offering limited support for technical repositories that combine documentation with other artifacts (e.g., source code) (Tao et al., 2025). To address the aforementioned challenges, this research presents a hybrid RAG system designed specifically for multi-source technical documentation. Our approach integrates heterogeneous artifacts, including PDFs, GitHub repositories, and Web content, within a *unified retrieval and generation pipeline* that supports cross-source synthesis of responses. Our pipeline employs different building blocks tailored to each format and preserves source metadata by means of techniques such as chunking, retrieval, re-ranking, and chain-of-thought generation, which enable an explicit attribution of provenance information in response to user's prompts. We evaluated our approach using a prototype tool based on `Langflow`[4]. With this tool, we performed a scenario-based assessment using an LLM-as-judge protocol across single- and multi-source technical queries. Our initial results indicate that hybrid RAG architectures can support technical knowledge retrieval across fragmented documentation.

2 Background

Retrieval-Augmented Generation (RAG) is a technique for improving the outputs of an LLM by means of external knowledge sources, which were unknown or unavailable to the LLM during the pre-training phase. In the context of software technical documentation, effective RAG systems require careful design choices across different dimensions, including retrieval mechanisms, document chunking strategies (e.g select a suitable vector database), re-ranking, code-aware processing, and evaluation methodologies (Gao et al., 2023).

RAG foundations and evolution. RAG was introduced in (Lewis et al., 2020) as a method to combine parametric knowledge encoded within LLMs with non-parametric knowledge retrieved from external corpora. Early RAG architectures followed a simple retrieve-then-generate pipeline, but subsequent work has extended this paradigm toward more flexible and adaptive designs. A recent survey (Gao et al., 2023) describes the evolution of RAG systems around three categories: *Naïve RAG*, which performs basic retrieval followed by generation; *Advanced RAG*, which incorporates retrieval optimization techniques such as query rewriting and filtering; and *Modular RAG*, which enables dynamic pipeline composition and component specialization. Our system aligns more closely with a *Modular RAG*, as it integrates specialized ingestion, retrieval, and ranking components tailored to different documentation sources and query scenarios.

In addition, the effectiveness of a RAG system is strongly influenced by how the source documents are segmented (chunking) and indexed. Recent research indicates that static, fixed-length chunking can split semantic concepts and introduce irrelevant context, lowering retrieval precision and completeness in RAG

[4] https://www.langflow.org/

systems (Bhat et al., 2025). This is challenging for structured and dense technical documentation where logical boundaries do not align with arbitrary segment sizes. Instead, chunking strategies that preserve semantic or structural boundaries (such as sentence-level segmentation followed by context expansion) are shown to improve retrieval performance compared to static fixed-size chunking, as they maintain conceptual coherence and reduce irrelevant context fragmentation (Wang et al., 2024). Our system employs syntactic chunking, preserving document metadata to support later traceability and cross-source synthesis.

Embedding-based retrieval using dual-encoding models often produces candidate contexts that are topically related with the user's query. However, the most relevant context might be lost in the middle of the retrieval process, and thus the generation might be inaccurate or lack precision. As a result, re-ranking has become a critical component in modern RAG pipelines to improve precision before generation (Liu et al., 2024). Re-ranking methods typically rely on cross-encoder architectures that jointly encode query–document pairs. Our system integrates a re-ranker to refine retrieved contexts prior to generation.

Understanding codebases with LLMs. While many RAG systems use documents as knowledge sources, technical documentation frequently comes together with source code repositories that introduce additional challenges. A primary difficulty is often understanding why specific code exists and how it connects to other artifacts across a fragmented repository. Code artifacts exhibit structural dependencies, implicit semantics, and cross-file relationships that are difficult to capture by text-based retrieval alone. Recent developments, such as *CodeWiki*[5] and *DeepWiki*[6], provide LLM-based mechanisms to generate structured documentation from code repositories, allowing users to navigate from high-level concept explanations to concrete artifacts. Recent benchmarks, such as *SWE-QA* (Peng et al., 2025), highlight the need for multi-hop reasoning across large codebases, while *CoReQA* (Chen et al., 2025) highlights the need for retrieval and generation systems that can navigate extensive repository context rather than isolated code snippets.

On the other hand, evaluating RAG systems poses challenges that extend beyond traditional information retrieval or text generation metrics. Standard NLP measures are not always adequate to capture whether the generated responses are based on retrieved sources (Salemi & Zamani, 2024). Several evaluation frameworks have been proposed to address these limitations. *RAGAS* (Es et al., 2024) introduces metrics for faithfulness, answer relevance, and context precision, while *FActScore* (Min et al., 2023) focuses on factual consistency with source documents. More recently, LLM-as-judge methodologies have gained adoption as scalable alternatives to manual annotation, provided they are carefully calibrated against human judgments[7]. Our evaluation strategy follows an LLM-as-judge strategy and scenario-based analysis of technical documentation.

[5] https://codewiki.google/
[6] https://deepwiki.com/
[7] https://www.evidentlyai.com/llm-guide/llm-as-a-judge

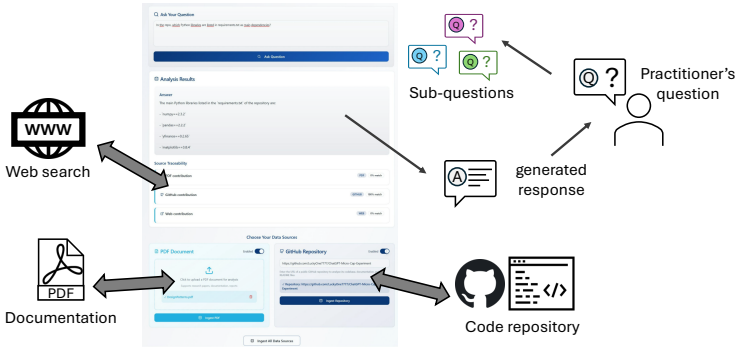

Fig. 1. High-level view of a Q&A system for technical documentation.

3 Motivating Example: Searching about a Design Pattern in a Repository

To illustrate the challenges of fragmented technical documentation mentioned in the previous sections, let us consider a software practitioner tasked with extending the *ChatGPT-Micro-Cap-Experiment* repository[8], which is an LLM-powered system for managing a real-money micro-cap stock portfolio. The practitioner needs to understand how the system implements different stock analysis logic in order to extend it.For example, the practitioner might start with the question *"How is the Strategy pattern used for stock analysis in this repository, and what steps are needed to add a new 'Technical Indicator' strategy?"*. From theoretical knowledge (e.g., a design book (Gamma et al., 1994)), the practitioner knows that the *Strategy Pattern* is ideal for swapping algorithms at runtime. However, this is not enough for the question, and additional issues need to be investigated.

- How is this pattern concretely implemented in the current codebase?
- Which specific files represent the "Context" and the "Concrete Strategies"?
- How to add a new analysis strategy (e.g., a "Sentiment Analysis Strategy") following the existing project conventions?
- Is the pattern implemented in conjunction with other related patterns?

In this context, the practitioner's question might need to be decomposed into sub-questions, and the she must manually perform the following steps: i) search a PDF textbook to refresh her memory on the *Strategy Pattern*'s structure, ii) browse the GitHub repository (e.g., 'analysts/' or 'processors/' directories) to find matching code structures, iii) check Web documentation (e.g., OpenAI API docs or financial library wikis) to ensure the new strategy's dependencies are up to date, and iterate until obtaining a satisfactory response.

This scenario exemplifies the nature of the *multi-source* reasoning process, where code, documentation, and domain knowledge need to be integrated but

[8] https://github.com/LuckyOne7777/LLM-Trading-Lab

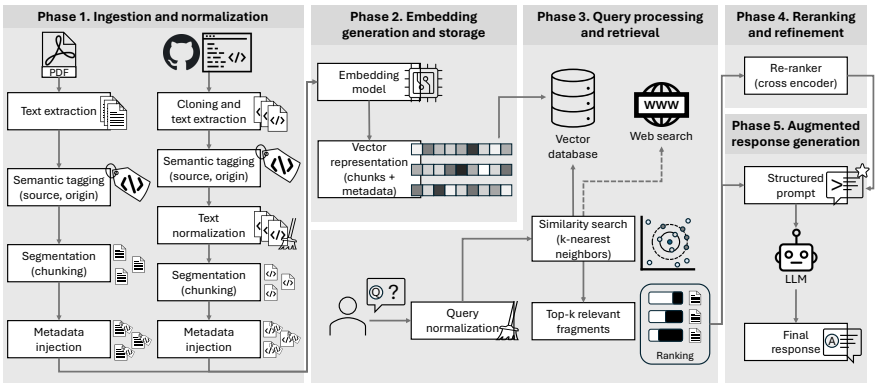

Fig. 2. Phases of the proposed hybrid RAG architecture.

still be *traceable* in the final response, as depicted in Figure 1. A *Naïve RAG* cannot handle this type of questions and sources. In constrast, a more advanced RAG can implement a workflow as follows: i) the system retrieves the definition of the *Strategy* pattern from the PDF textbook, ii) it identifies the 'BaseAnalyst' class and its implementations in the GitHub repository (e.g., 'FundamentalAnalyst.py'), iii) it performs a Web search to fill in gaps with best practices (for the pattern or for the code base), and finally iv) it returns a synthesized answer. In addition, it can explicitly cite the PDF for the pattern's intent and the specific GitHub implementation, reducing the practitioner's cognitive load of switching between heterogeneous sources.

4 Technical Approach

We propose a hybrid RAG approach that is guided by three principles. First, it must *support effective semantic retrieval across heterogeneous documentation sources*. These sources can include PDFs representing theoretical material, GitHub repositories capturing implementation artifacts, and Web content providing up-to-date information, all accessible through a unified query interface. Second, the system must *preserve traceability*, maintaining provenance metadata from ingestion through response generation to enable explicit citation of source fragments. Third, *hallucinations must be minimized* by constraining the LLM output to the retrieved context.

Figure 2 depicts the architecture of our RAG approach,which works in five sequential phases: i) ingestion and normalization, ii) embedding generation and vector storage, iii) query processing and retrieval, iv) re-ranking and refinement, and v) augmented response generation. Our RAG prototype was implemented using *Langflow*. All code, benchmarks, and deployment instructions are available at: https://anonymous.4open.science/r/HybridRAG-F6C8

Phase 1: Ingestion and normalization. The system implements parallel ingestion pipelines tailored to different source types. PDF documents are processed using

`PyPDF2`[9] for text extraction, followed by semantic tagging with document-level metadata such as filename and source type. GitHub repositories are processed through a dedicated `GitExtractor` component, which clones repositories and iterates over relevant source files. A design decision here is the explicit injection of filename context into chunk content, for example, by prefixing code segments with identifiers. This strategy improves retrieval performance for file-specific queries and supports architectural traceability across repository artifacts. All ingested documents are mapped to a unified metadata schema shared across sources. Each chunk records its origin (PDF document, GitHub repository), a concrete source identifier such as a filename, and a timestamp corresponding to the ingestion event.

Phase 2: Embedding and vector storage. After normalization, document chunks are embedded using NVIDIA's `nv-embed-v1` model[10], which produces 768-dimensional representations. Preliminary comparisons showed improved performance on technical and code-related content relative to general-purpose embedding models. We defined a chunk range between 512 and 1024 tokens with a 128-token overlap, balancing semantic density with contextual continuity. Embedded representations are stored in `ChromaDB`[11] using cosine similarity.

Phase 3: Query processing and retrieval. User queries undergo the same preprocessing pipeline as document content, including normalization and embedding generation. The system performs semantic retrieval in parallel across all enabled source collections. Conditional routing mechanisms allow individual sources (PDFs, GitHub repositories, or Web content) to be activated based on frontend configuration. Dynamic web knowledge is integrated as a complementary retrieval source. Web search acts as a fallback mechanism: when local sources such as PDFs or GitHub repositories do not provide sufficient information, the system automatically queries the web to obtain up-to-date data. For Web search, we relied on the `Jigsaw API`[12]. The output of this phase consists of the top-ranked candidate chunks per source, which are forwarded to the refinement stage.

Phase 4: Re-ranking and refinement. To improve retrieval precision, the system applies cross-encoder-based re-ranking to the initially retrieved results. The top-10 chunks from each source are passed to NVIDIA's `llama-3.2-nv-rerankqa-1b-v2` model[13], which jointly encodes query–document pairs and assigns relevance scores on a 0˘10 scale. Our pipeline explicitly preserves chunk metadata throughout the re-ranking process to ensure provenance information.

Phase 5: Augmented response generation. The highest-ranked chunks are assembled into a structured prompt that explicitly constrains the LLM to use

[9] https://github.com/py-pdf/pypdf
[10] https://build.nvidia.com/nvidia/nv-embed-v1
[11] https://www.trychroma.com/
[12] https://jigsawstack.com/
[13] https://build.nvidia.com/nvidia/llama-3_2-nv-rerankqa-1b-v2

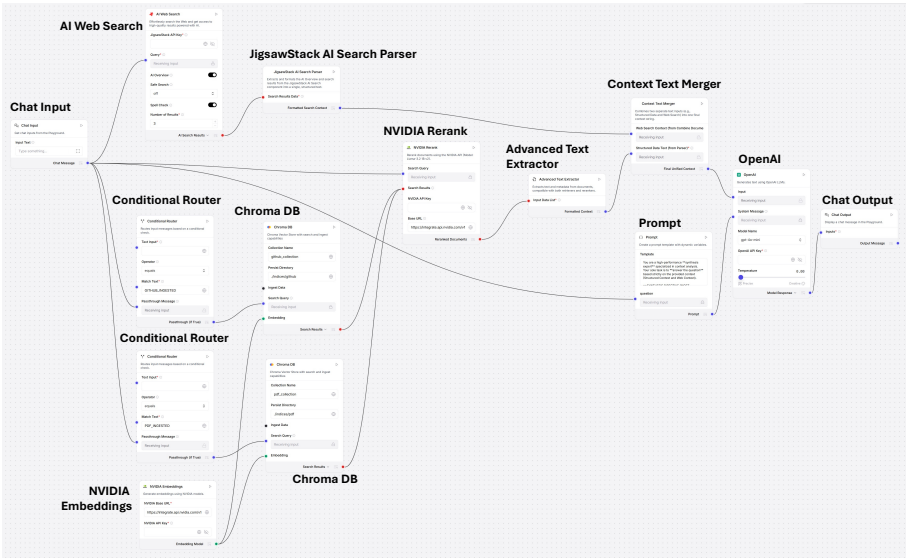

Fig. 3. Excerpt of the generation flow of our hybrid RAG, as defined in *Langflow*. The execution proceeds from left to right.

only retrieved context. The prompt includes both textual content and associated metadata, allowing the LLM to reason over heterogeneous sources while maintaining attribution. For our experiments, we started with an initial prompt which was iteratively improved until satisfactory results were obtained for a question sample. Responses were generated using `GPT-4o-mini` with the temperature set to 0, which reduces output variability and limits non-deterministic behavior[14]. This configuration supports controlled benchmarking across evaluation scenarios.

`Langflow` prototype. To implement the proposed architecture, we developed a prototype using `Langflow`, which is a low-code, visual platform for building multi-agent and RAG applications. `Langflow` follows a modular strategy in which the user can drag and drop components as building blocks to construct workflows, allowing for the seamless integration of various LLMs and vector databases. The RAG orchestration is achieved by connecting specialized components for ingestion, retrieval, and generation. We leveraged the platform extensibility to incorporate custom components for tasks, such as metadata-aware re-ranking and format-specific normalization. Figure 3 shows an excerpt of the generation part of our RAG.

[14] https://platform.openai.com/docs/api-reference/chat

Table 1. Benchmark Query Distribution and Examples

Scenario	Example Query	Number
PDF-only (theoretical)	"What is the intent of the Singleton pattern?"	7
GitHub-only (code)	"Which Python libraries are listed in requirements.txt?"	7
Combined Sources	"Apply Decorator pattern to add logging to ingestion pipeline."	7
Web-only (general)	"Latest stable Python version as of October 2025?"	6

5 Evaluation

The main goal is to assess the effectiveness of our hybrid RAG system. To this end, we took a GitHub repository (the one used in the example of Section 3) and built a (small) benchmark of 27 technical pairs (query-response) to reflect usage scenarios requiring technical documentation. The benchmark spans four complementary settings: *theoretical* questions answerable from PDF documentation, *repository-level* questions targeting source, queries requiring *synthesis across multiple sources*, and *general technical* questions to be supported by Web content. Table 1 summarizes the distribution of queries and provides representative examples for each scenario.

For each query, we defined an expected answer that was verified against the original documentation by an expert. In addition, we added the ideal source attribution distribution indicating the intended contribution of each source type (e.g., PDF: 70%, GitHub: 30%, Web:0%), and a complexity label distinguishing simple fact retrieval (simple), single-source synthesis (moderate), and multi-source reasoning tasks (complex).

Evaluation was performed using an LLM-as-judge protocol (Zheng et al., 2023) with *GPT-4.1* as the evaluator (temperature was set to 0). To validate reliability, a subset of responses was manually inspected by the authors and used for calibration. Each query was executed 10 times to assess output stability. Four evaluation metrics were assessed on a 1˘10 scale (with 10 being the maximum score and 1 the worst one). These metrics were the following:

- **Content Accuracy.** Factual correctness and semantic equivalence to expected answer.
- **Completeness.** Coverage of key concepts and points.
- **Source Attribution Accuracy.** Alignment between reported and ideal source contributions.
- **Conciseness.** Absence of redundant or irrelevant information.

The average and standard deviation values of the metrics are given in Table 2. The system consistently achieved high content accuracy (8.7) with low variance, indicating stable factual correctness in repeated executions. Completeness followed a similar trend, while attribution accuracy (7.5) exhibited greater

Table 2. Overall System Performance (Scale: 1-10)

Metric	Mean	Std Dev
Content Accuracy	8.72	0.10
Completeness	8.52	0.18
Source Attribution Accuracy	7.47	0.22
Conciseness	7.36	0.15

variability, particularly for the queries requiring synthesis across multiple documentation sources.

Performance varied across query scenarios. *PDF-only queries* achieved near-perfect performance, confirming effective semantic retrieval from structured technical documents. *GitHub-only queries* maintained strong content accuracy but show sometimes lower attribution scores, mainly due to confusion between repository artifacts and conceptually similar PDF sources. *Combined-source queries* presented the greatest challenge. Although answers were typically correct, the RAG system often struggled to quantify relative source contributions. *Web-only queries* performed consistently well, demonstrating appropriate fallback behavior when local documentation was insufficient.

For the reference queries, the average end-to-end latency of the system was 18.4 seconds ($\pm$ 3.2 seconds), which is very reasonable for RAG systems. Retrieval and embedding (`ChromaDB` + embedding) account for approximately 2.1 seconds, re-ranking (NVIDIA cross-encoder) for 4.8 seconds, Web search (when triggered, `JigsawStack API`) for 8.2 seconds, and response generation (`GPT-4o-mini`) for 3.3 seconds. These results evidence the usual tradeoff between quality of the results and processing time in RAG systems.

The evaluation highlighted several findings. First, injecting metadata directly into chunk content (e.g., filename information for repository artifact) substantially improves file-specific retrieval accuracy. Second, re-ranking turned out to be a critical component for multi-source queries, filtering lexically similar but contextually irrelevant chunks and thus improving attribution quality.

On the downside, the experiments exposed a number of limitations. First, our system does not yet incorporate structure-aware code representations, limiting performance on queries requiring cross-file reasoning or architectural analysis. Second, visual information in PDFs, such as diagrams and tables, is lost during text extraction, limiting comprehension of visual-dependent concepts. Third, the current chunking strategy is static and does not adapt to higher-level document structure boundaries (e.g., chapters, functions, modules). Finally, even when synthesized response was correct, the system tended to exhibit a bias towards text-dense sources such as PDFs, underestimating code contributions, which suggests the need for better attribution mechanisms.

5.1 Threats to Validity

Despite the promising results revealed by our empirical analysis, we identified several threats to internal, construct and external validity.

Internal validity. The integration of multiple components (including retrievers, re-rankers, and prompt templates) introduces potential interactions in the RAG pipeline that are difficult to isolate fully. Although ablations can be performed during development, not all combinations can be exhaustively evaluated. In addition, the stochastic behavior in LLMs may influence response quality. We mitigated this threat by setting the temperature to 0 and repeating each query multiple times, but variability in the pipeline behavior cannot be ruled out.

Construct validity. Our study relies primarily on LLM-as-a-judge to assess answer quality, completeness, conciseness, and source attribution. Although this approach enables scalable evaluation, it may not fully reflect human judgment, particularly for nuanced architectural reasoning or partial correctness. To mitigate this threat, we calibrated the evaluator against a manually annotated subset. Source attribution accuracy presents an additional construct challenge. The reported attribution scores depend partly on the LLM ability to self-report source contributions, which may not precisely reflect the true influence of retrieved contexts. While ideal attribution distributions were defined for each query, these targets remain approximations and may oversimplify complex synthesis processes involving overlapping information across sources.

External validity. Our benchmark consists of 27 constructed technical queries spanning PDFs, a GitHub repository, Web content, and cross-source scenarios. While designed to reflect realistic usage patterns, the benchmark represents only a subset of possible technical documentation tasks for a single repository. This does not cover all programming languages, repository structures, or documentation styles. As a result, the reported findings should be interpreted as initial trends rather than general ones.

6 Related work

Recent research has explored different strategies to enhance RAG systems by integrating heterogeneous information sources. A common objective across these works is to mitigate hallucinations and improve response grounding through richer contextual evidence. Some approaches focus on improving retrieval quality through hybrid ranking and context modeling. For instance, *HF-RAG* (Santra et al., 2025) combines labeled and unlabeled contexts to estimate both topic-specific claim likelihood and contextual relevance. Their approach employed multiple ranking strategies and demonstrated improved performance over a verification against textual source task. While effective, the approach primarily targets textual document collections and does not address repository-based artifacts.

Another research direction combines RAG with structured knowledge representations. In this context, in (Sarmah et al., 2024), the authors introduced a Hybrid-RAG that integrates knowledge graphs with document retrieval to support context-aware response generation in the financial domain. Similarly, in (Wan et al., 2025) an architecture that unifies vector-based and graph-based retrieval over domain-specific corpora was proposed. These approaches demonstrated improved factual consistency, although relying on explicit knowledge

graphs that must be constructed in advance. This kind of graphs may be unavailable or costly to maintain for many software engineering artifacts.

A complementary line of work investigates the relationship between RAG and long-context language models (LC-LLMs). In this context, in (Z. Li et al., 2024), the authors compared RAG-based systems with LC-LLMs capable of processing large context windows, as well as hybrid combinations of both approaches. Their results indicated that while RAG and LC-LLMs often achieved comparable performance, LC-LLMs can outperform retrieval-based methods when relevant context is missed, albeit at substantially higher token and computational costs. To address this trade-off, a self-routing strategy that dynamically selected between retrieval and long-context prompting was proposed. In (X. Li et al., 2025), failure modes of RAG and LC-LLM approaches across various datasets were analyzed, highlighting the conditions under which each strategy is more effective.

Collectively, these works illustrate the ongoing evolution of hybrid RAG architectures that combine retrieval paradigms, ranking mechanisms, and contextualization strategies to improve answer quality. To the best of our knowledge, existing approaches primarily focus on homogeneous document collections or structured knowledge sources. In contrast, our work addresses a distinct gap by integrating unstructured documents and software repositories within a RAG pipeline, while preserving traceability across heterogeneous technical artifacts.

7 Conclusions

In this paper, we presented a hybrid RAG system for querying multi-source technical documentation spanning PDFs, GitHub repositories, and Web content. The proposed architecture demonstrates that the integration of retrieval, re-ranking, and generation components enables accurate and verifiable responses to technical queries that span across heterogeneous artifacts. This work contributes to enhance technical documentation access, supporting both learning and professional software engineering practice.

Our experimental results, although initial, show that documentation and implementation-related resources can be effectively unified within a single retrieval pipeline. By preserving provenance metadata throughout the pipeline, the system supports traceability from generated responses back to their original sources, which is an essential requirement in professional engineering contexts. To demonstrate the feasibility of the approach, we developed a prototype RAG system that provides a functional and extensible foundation for intelligent technical documentation assistants, bridging fragmented knowledge sources and supporting integrated technical understanding.

Our findings also highlight several promising directions for future work. First, we plan to incorporate code-aware retrieval mechanisms (such as abstract syntax tree–based chunking, function-level embeddings, and call-graph context) could substantially improve repository question answering. Second, more advanced multi-hop or graph-based RAG strategies could enable iterative information gathering for complex cross-source queries. For instance, recent work, such as *RankRAG* (Yu et al., 2024), explores a tighter integration between ranking and

generation. Third, the pipeline could be extended with multi-modal processing, which would allow the inclusion of diagrams and figures commonly found in technical PDFs. Finally, we will explore independent attribution mechanisms decoupled from LLM generation, and lightweight query-routing classifiers capable of selecting optimized paths based on query intent.

References

Bhat, S. R., Rudat, M., Spiekermann, J., & Flores-Herr, N. (2025). Rethinking chunk size for long-document retrieval: A multi-dataset analysis.

Chen, J., Zhao, K., Liu, J., Peng, C., Liu, J., Zhu, H., Gao, P., Yang, P., & Deng, S. (2025). Coreqa: Uncovering potentials of language models in code repository question answering.

Daka, E., & Fraser, G. (2014). A survey on unit testing practices and problems. *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, 201–211.

Es, S., James, J., Anke, L. E., & Schockaert, S. (2024). Ragas: Automated evaluation of retrieval augmented generation. *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - System Demonstrations, St. Julians, Malta, March 17-22, 2024*, 150–158.

Forward, A., & Lethbridge, T. (2002). The relevance of software documentation, tools and technologies: A survey. *Proceedings of the 2002 ACM Symposium on Document Engineering, McLean, Virginia, USA, November 8-9, 2002*, 26–33.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley Professional.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Guo, Q., Wang, M., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *CoRR*, *abs/2312.10997*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks.

Li, X., Bai, Y., Jin, B., Zhu, F., Pan, L., & Cao, Y. (2025). Long context vs. rag: Strategies for processing long documents in llms. *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 4110–4113.

Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024). Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 881–893.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, 157–173.

Min, S., Krishna, K., Lyu, X., Lewis, M., Yih, W., Koh, P. W., Iyyer, M., Zettlemoyer, L., & Hajishirzi, H. (2023). Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *Proceedings of the 2023 EMNLP 2023*, 12076–12100.

Peng, W., Shi, Y., Wang, Y., Zhang, X., Shen, B., & Gu, X. (2025). SWE-QA: can language models answer repository-level code questions? *CoRR*, *abs/2509.14635*.

Robillard, M. P., Marcus, A., Treude, C., Bavota, G., Chaparro, O., Ernst, N., Gerosa, M. A., Godfrey, M., Lanza, M., Linares-Vásquez, M., et al. (2017). On the role of information diversity in software documentation. *IEEE Software*, *34*(6), 74–80.

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

Santra, P., Ghosh, M., Ganguly, D., Basuchowdhuri, P., & Naskar, S. K. (2025). HF-RAG: hierarchical fusion-based RAG with multiple sources and rankers. *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM 2025, Seoul, Republic of Korea, November 10-14, 2025*, 5202–5207.

Sarmah, B., Mehta, D., Hall, B., Rao, R., Patel, S., & Pasquali, S. (2024). Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. *Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF 2024, NY, USA, November 14-17, 2024*, 608–616.

Tao, Y., Qin, Y., & Liu, Y. (2025). Retrieval-augmented code generation: A survey with focus on repository-level approaches. *arXiv:2510.04905*.

Wan, Y., Chen, Z., Liu, Y., Chen, C., & Packianather, M. (2025). Empowering llms by hybrid retrieval-augmented generation for domain-centric q&a in smart manufacturing. *Adv. Eng. Inform.*, *65*(PB).

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., Lv, C., Zheng, X., & Huang, X. (2024, November). Searching for best practices in retrieval-augmented generation. In Y. Al-Onaizan, M. Bansal, & Y.-N. Chen (Eds.), *Proceedings of the 2024 conference on empirical methods in natural language processing* (pp. 17716–17736). Association for Computational Linguistics.

Yu, Y., Ping, W., Liu, Z., Wang, B., You, J., Zhang, C., Shoeybi, M., & Catanzaro, B. (2024). Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *NeurIPS 2024, Vancouver, BC, Canada*.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *Proceedings of the 37th International Conference on Neural Information Processing Systems*.