ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CESENA CAMPUS

Department of Computer Science and Engineering DISI
Second-Cycle Degree in Digital Transformation Management
Class: LM-91

Building and Evaluating a Document-Grounded QA System: A RAG-Based Approach

Graduation thesis in:
Data Mining and Machine Learning

Supervisor
Prof. Matteo Francia

Candidate
Alessandro De Faveri

Session V
Academic year 2025-2026

Abstract

Large Language Models (LLMs) are increasingly used to support study and research tasks, yet their adoption in academic settings is constrained by the need for verifiability and traceability. In particular, when answering questions about scientific papers, responses must be grounded in primary sources and should enable readers to locate the supporting evidence, ideally at page level. This thesis addresses the problem of question answering over a local corpus of faculty-authored papers that are not reliably covered by an LLM's training data, aiming to improve correctness while providing citations that support rapid validation.

To this end, the thesis designs and implements an end-to-end Retrieval-Augmented Generation (RAG) system for scientific PDFs. The pipeline performs page-level PDF text extraction, punctuation-aware chunking with overlap (chunk size 1000, overlap 200), embedding computation using `all-MiniLM-L6-v2`, and indexing in Qdrant with provenance metadata (`source`, `page`). At query time, the system retrieves the top-k most similar chunks (`TOP_K`=5), constructs a prompt that injects retrieved evidence, and generates an answer using either local LLM backends via Ollama (`llama3.2`, `qwen2.5:7b`) or an optional cloud backend. Prompt engineering is treated as a first-class design variable through five prompt templates and an optional open-knowledge mode.

The evaluation is conducted on an internal benchmark of 10 questions with reference answers and expected provenance. Results show that prompt design significantly affects answer alignment: the strict template (T4) achieves the highest average similarity (0.613) across backends, outperforming more permissive templates. Performance varies by question category: in-scope queries obtain the highest similarity (up to 0.738 with `qwen2.5:7b`), while out-of-scope queries remain challenging. Overall, the work demonstrates that combining dense retrieval, provenance-aware indexing, and carefully designed prompts can improve controllability and traceability for academic paper question answering, and it outlines future directions focused on prompt optimisation and decomposition-driven prompting for complex multi-document queries.

Optional. Max a few lines.

Contents

List of Figures

List of Listings

Chapter 1

Introduction

1.1 LLMs and RAG

In recent years, Large Language Models (LLMs) have become widely adopted tools in activities related to study and research, including literature review support, summarisation of technical material, drafting, technical translation, and, more generally, *question answering* over document collections. Despite their effectiveness in producing fluent and coherent text, the quality of an LLM answer depends critically on the availability of relevant and verifiable knowledge at generation time. General-purpose models may produce plausible responses even when adequate evidence is missing, or when a question requires highly specific details that are only contained in a local corpus (e.g., the scientific papers produced within a research group). In academic contexts, this limitation is particularly problematic, since it can lead to either generic answers that fail to capture the required specificity, or to unsupported statements (often referred to as *hallucinations*)that are difficult to detect and validate.

In scientific practice, the *traceability* of information is not an optional feature but a methodological constraint: a claim is considered acceptable if it can be connected to an explicit, consultable source. This requirement is inherently in tension with the native operating mode of general LLMs (see the schematic comparison in fig. 1.1). While a model may encode broad domain knowledge, it does not necessarily cover the content of a specific local corpus, nor can it guarantee that each

detail is correct and attributable to a precise passage. As a result, using LLMs without explicit grounding mechanisms in a university setting risks producing answers that are hard to verify and potentially incorrect, undermining trust in the overall document consultation process.

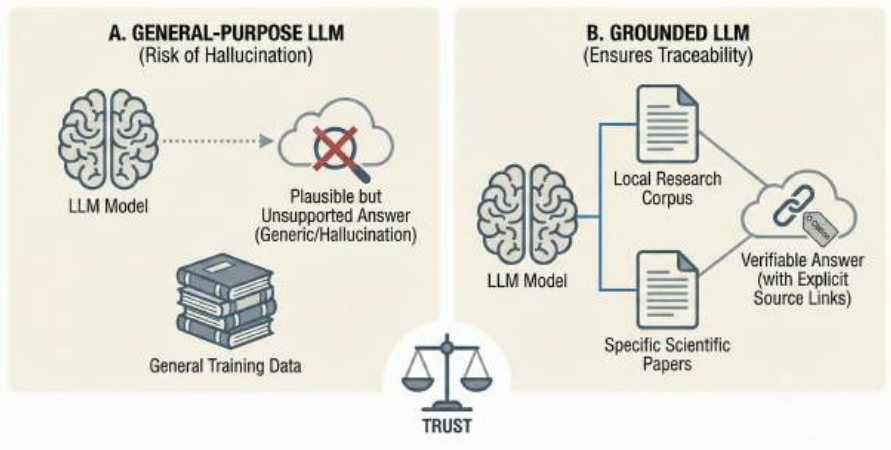

Figure 1.1: General Purpose LLM vs. Grounded LLM

A widely adopted approach to mitigate these limitations is Retrieval-Augmented Generation (RAG), which combines a retrieval component over an external document base with a generative model that uses retrieved passages as context. The intent is to reduce reliance on the model's internal knowledge and constrain generation to explicit evidence. However, both experience and recent literature show that adding retrieval does not automatically guarantee improvements. First, answer quality depends on the quality of retrieved passages. If irrelevant or misleading content is included in the context, the model may be diverted from correct evidence and generate worse outputs than a configuration without retrieval. This phenomenon has been highlighted in benchmarks that study RAG robustness under misleading evidence, showing that several RAG systems can underperform a zero-shot baseline when the context contains adversarial or contradictory passages. [1]

Second, there is a further constraint related to *context length*. The intuitive assumption that "more context yields better answers" is not always true. Recent studies suggest that even when the correct evidence is present, reasoning perfor-

mance can degrade as the input becomes longer. In other words, providing additional text does not guarantee effective utilisation of evidence, and long-context settings may reduce the model's ability to isolate and apply relevant information. [2] This observation is particularly relevant for academic paper: scientific PDF papers are dense and structured, and they can quickly produce large volumes of text. A system aimed at answering *vertical* questions over such documents must therefore balance coverage of evidence with control over noise and redundancy in the retrieved context.

Within this framework, this thesis addresses a concrete need: given a set of scientific papers from a specific research area (e.g., those produced by a university research group), the goal is to design and implement a RAG system that enables an LLM to answer domain-specific questions about such papers even when the model has not been trained on them. In this scenario, precision is a primary requirement. Scientific papers include definitions, assumptions, methodological details, and experimental results that do not tolerate approximations. Consequently, the objective is not merely to produce plausible answers, but to systematically reduce errors and support each claim with documentary evidence.

1.2 Objectives and Approach

This work focuses on four main dimensions. The first dimension is *robustness*: how does the system behave when queries are ambiguous, incomplete, or *non-answerable* given the available corpus? In an academic setting, this implies distinguishing between questions supported by explicit evidence, questions supported only partially (requiring cautious answers and explicit uncertainty), and out-of-scope questions where invented content would be misleading.[1] The second dimension concerns *design choices*: the work evaluates how performance and fidelity to the local corpus change as key design parameters vary, including chunking strategies, retrieval parameters such as *top-k*, prompt templates, and operational modes. [3] The third dimension is *grounding and citation quality*: beyond answer correctness, the system is assessed on whether provided citations refer to the correct documents and precise locations (e.g., page references) that support the generated statements. Recent work on LLM-assisted academic writing treats citation cor-

rectness as a distinct target, rather than a by-product of fluent generation. [4] Finally, the fourth dimension concerns *answer quality*: the work aims to quantify improvements not only relative to a no-retrieval baseline, but also across different prompting strategies, since prompt design can materially influence whether the model overgeneralises, "fills in" missing evidence, or adheres to stricter grounding instructions.

To address these objectives, the thesis adopts a structured RAG pipeline consisting of document preprocessing, segmentation into retrievable units (chunking), embedding computation and indexing in a vector store, retrieval of relevant passages given a user query, answer generation conditioned on the retrieved context, and experimental evaluation. The evaluation aspect is central: RAG performance depends on several interacting variables, and the effect of any single modification is difficult to isolate without systematic experimentation and structured reporting. Recent benchmarks also show that RAG systems exhibit varying degrees of resilience depending on query type and the nature of the reasoning required, reinforcing the need to evaluate not only overall accuracy but also robustness to non-ideal contexts. An additional methodological aspect of this thesis is the explicit treatment of *prompt engineering* as a design variable. Small changes in prompt structure can significantly affect model behaviour, including its tendency to introduce unsupported details, its style of attribution, and its willingness to return abstentions when evidence is weak. For this reason, prompt templates are not treated as superficial interface details, but as first-class components that contribute to the controllability and reliability of the overall system.

The main contribution of this work is the design and implementation of an end-to-end system for querying an academic PDF corpus with emphasis on grounding. First, a document indexing pipeline is developed to preserve metadata that enable traceable linking from retrieved text to the source documents. Second, alternative retrieval and generation strategies are implemented and compared, including citation-oriented constraints and two operational modes reflecting the trade-off between corpus fidelity and general knowledge integration. Third, an experimental protocol is defined using an internal benchmark dataset and evaluation metrics targeting both answer quality and citation accuracy, enabling reproducible comparisons across configurations. Finally, the work provides an empirical analysis

of prompt engineering by evaluating multiple prompt templates under controlled conditions and quantifying their effect on correctness and grounding.

1.3 Thesis Structure

The remainder of this thesis is organised as follows. Chapter 2 introduces the application context and refines the problem statement in the specific setting of academic paper analysis. It describes the target corpus of PDF papers, the requirements for traceability and citation-level verification (including page-level references), and the constraints that motivate a retrieval-augmented approach when the underlying LLM has not been trained on the target documents.

Chapter 3 reviews the relevant literature on Retrieval-Augmented Generation, focusing on the design choices that most directly affect factuality and traceability in document-grounded question answering. In particular, it discusses document preprocessing and chunking strategies, embedding-based retrieval and vector databases, grounding-oriented prompting strategies, and evaluation methodologies for answer quality and citation correctness in academic writing and information-seeking tasks.

Chapter 4 presents the proposed system end-to-end and details the engineering contribution. It describes the ingestion pipeline that processes PDFs and builds the index with traceability metadata, the embedding and retrieval layer built on a vector database, and the generation layer that supports both local and cloud LLM backends. Specifically, the chapter explains how the system integrates Qdrant as the vector store, uses Ollama as a local LLM server, and optionally relies on Azure OpenAI models when credentials are provided. It also formalises the two operating modes adopted in this work, namely a corpus-only mode that enforces document grounding and an open-knowledge mode that allows the model to integrate general knowledge, and it clarifies how prompt templates are treated as first-class, configurable components of the pipeline.

Chapter 5 describes the experimental methodology and reports the results. It introduces the internal benchmark dataset used for evaluation, the execution protocol that systematically explores combinations of LLM backends, prompt templates,

retrieval parameters (including top-k), and operating modes, and the metrics used to quantify both answer quality and grounding. The chapter also discusses the evaluation workflow implemented in the project, including structured result export to CSV for reproducible comparisons and detailed logging for error analysis at the level of retrieved sources, cited pages, and generated responses.

Finally, Chapter 6 summarises the main findings and discusses limitations and future directions. It highlights how system behaviour changes across configurations, with particular attention to prompt engineering as a controllability lever, and outlines possible extensions such as improved PDF parsing for complex layouts, more advanced reranking strategies, and additional mechanisms to increase robustness to ambiguous or non-answerable queries while preserving traceability.

Chapter 2

Case study

2.1 RAG Pipeline Overview

Retrieval-Augmented Generation (RAG) denotes a family of approaches that combine information retrieval over an external knowledge base with neural text generation. Instead of relying exclusively on the parametric knowledge encoded in a language model, a RAG pipeline retrieves passages that are relevant to a user query and provides them as context for answer generation, thereby enabling explicit grounding on an updatable document collection. Figure 2.1 summarises the main stages of a typical RAG workflow. First, documents are processed and encoded into dense vector representations using an embedding model, then indexed in a vector database. At query time, the user question is encoded with the same embedding model and used to perform similarity search over the indexed representations. The retrieved passages are finally combined with the query to form a prompt that conditions the language model generation, producing a response that can be traced back to the underlying sources.

In academic document question answering, grounding is not only a mechanism to improve topical relevance: it is also a prerequisite for traceability. By associating metadata with each retrievable unit (e.g., document identifiers and page references), a RAG system can connect generated statements to verifiable sources and support a workflow in which users inspect the original passages that justify the answer.[4] In this sense, RAG provides a bridge between natural language in-

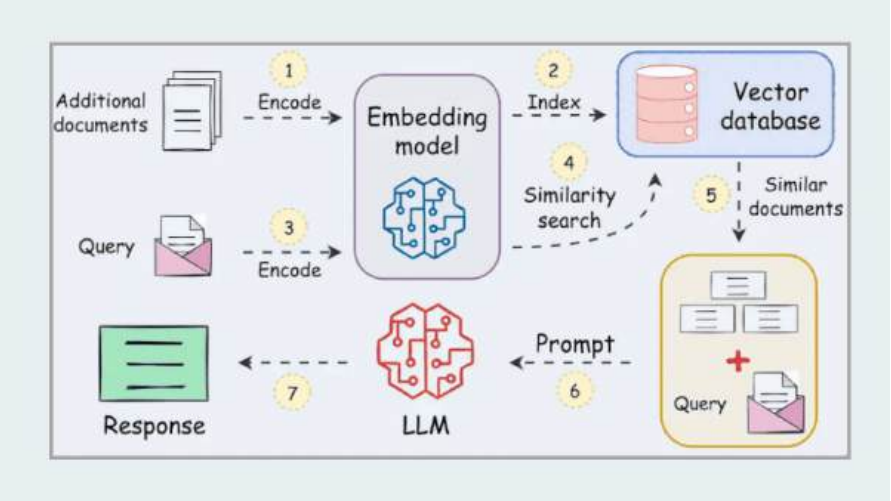

Figure 2.1: The main stages of a typical RAG pipeline.
Source: [5]

teraction and scholarly practices, where the value of an answer depends on the possibility of validating claims against primary material.

Designing an effective RAG pipeline requires addressing several interacting choices that directly affect answer reliability. A first aspect concerns how documents are segmented into retrievable units. Scientific papers are long and information-dense, and they must be partitioned into chunks that preserve enough local context for correct interpretation while remaining sufficiently focused to enable accurate retrieval. A second aspect concerns retrieval configuration, including how many passages are retrieved (e.g., top-k) and how they are ranked. Retrieving too few passages risks omitting critical evidence, whereas retrieving too many increases noise and may degrade generation by exposing the model to irrelevant or contradictory content.[1] A third aspect involves the interaction between retrieved evidence and prompting strategies: the model must be instructed to use the provided context appropriately, to avoid introducing unsupported details, and to preserve consistency with the retrieved sources.[6]

Importantly, "more context" does not necessarily imply better performance. Even when relevant evidence is present in the input, long-context settings may reduce the model's ability to locate and utilise it effectively, and performance can depend on where the relevant information appears within the context window.[2] This

observation further motivates careful context selection and the adoption of evaluation protocols that explicitly measure not only answer quality but also grounding and citation correctness.

For these reasons, in this thesis RAG is adopted as the core paradigm for question answering over a local corpus of scientific papers. The goal is to support traceable, evidence-based answers and to enable systematic comparisons across alternative configurations, including retrieval parameters, chunking strategies, prompt templates, and operating modes that trade off corpus fidelity and general-knowledge integration.[1]

2.2 System Context: Traceability Requirements in Academic Workflows

The case study addressed in this thesis is situated in an academic environment where the target knowledge base consists of scientific papers produced within a specific research context. In such settings, users typically seek precise information: definitions used by the authors, methodological choices, assumptions, experimental setups, or the interpretation of reported results. These information needs differ from generic web-style questions because they often require grounding in the exact wording of a paper and may involve subtle distinctions that are lost if the answer is based only on general domain knowledge.

A central requirement in this context is traceability. Unlike casual question answering, academic usage demands that key statements be verifiable and attributable to sources. When a language model produces an answer, the reader must be able to confirm whether the claims are supported by the referenced paper and to locate the relevant part of the document efficiently. Without this capability, even correct answers may be of limited utility, since they cannot be validated or used as a starting point for deeper reading. Conversely, plausible but incorrect answers are particularly harmful, as they can propagate misunderstandings and obscure the original contribution of the authors. This requirement highlights a fundamental tension between two desirable properties of a system that uses language models. On the one hand general-purpose models provide broad coverage

and can offer helpful explanations and background knowledge.

On the other hand, when the goal is to answer questions about a local corpus, fidelity to the specific documents becomes critical. A model that relies on general knowledge may produce answers that sound correct but do not reflect what is actually stated in the papers. For this reason, the ability to constrain generation to evidence from the corpus, and to explicitly signal when evidence is insufficient, is essential for reliability.

RAG-based systems are therefore a natural fit for this setting. By retrieving relevant passages from the paper collection and conditioning generation on them, the system can reduce the likelihood of unsupported content and provide the user with a transparent link between answer and evidence. Nevertheless, achieving this behaviour in practice requires careful problem formulation and requirement definition, since retrieval quality, document segmentation, and prompting strategies jointly determine whether the final output is accurate and traceable. The remainder of this chapter refines the case study by describing the target corpus, the user scenarios, and the functional and non-functional requirements that guide the design and evaluation of the proposed system.

2.3 Data Context: Faculty Paper Corpus and PDF-specific Challenges

The target knowledge base considered in this thesis consists of a local corpus of scientific publications authored by faculty members of the university. The corpus includes fifty full-length papers written in English, with a typical length between ten and twenty pages. This composition is representative of a realistic academic scenario in which the relevant information is concentrated in a bounded but non-trivial set of documents, and where users expect the system to answer questions that are closely tied to the specific content and terminology used by the authors.

From a structural perspective, full scientific papers exhibit a relatively stable macro-organisation, usually including an abstract, an introduction that frames the problem and contributions, a background or related work section, a methods or system description section, an experimental evaluation, and a conclusion. Al-

though this structure is beneficial for human readers, it poses design implications for a retrieval-based system. Questions asked by users may target different parts of a paper, ranging from high-level contributions to fine-grained experimental settings. As a consequence, the system must support retrieval not only of "the paper as a whole" but of localised passages that contain the evidence required to answer the question. This requirement motivates the need to represent documents as collections of retrievable units while preserving links to their original location in the source paper.

The length of the documents further amplifies this need. With papers in the ten-to-twenty page range, providing entire documents as context to a language model is typically impractical due to context-length constraints and the risk of diluting relevance. Instead, the system must select and provide a limited number of passages. This selection problem is non-trivial: scientific text is dense, and key information may depend on surrounding context such as definitions, assumptions, or references to earlier sections. Therefore, the document representation must balance granularity and completeness, enabling retrieval of specific evidence while preserving enough local context to support correct interpretation.

In addition to textual density, the corpus inherits common characteristics of PDF-based scientific writing. PDF documents are optimised for visual consumption and often include elements that are not purely linear text, such as figures, tables, equations, and references. Even when the primary goal is question answering over textual content, these elements influence how information is expressed and cross-referenced. For instance, experimental results may be summarised in tables, methodological details may be partially encoded in figure captions, and key statements may reference numbered sections or figures. These aspects introduce challenges for extraction and segmentation, and they motivate explicit assumptions about which parts of the paper are considered within scope for the system and how traceability will be maintained. Figure 2.2 summarises the main sources of complexity in scientific PDFs (e.g., multi-column layouts, tables, figures, and inconsistent encodings) that directly affect text extraction and downstream retrieval quality.

Finally, the fact that the corpus is local and authored by the university faculty has two important implications for the case study. First, it provides a coherent and

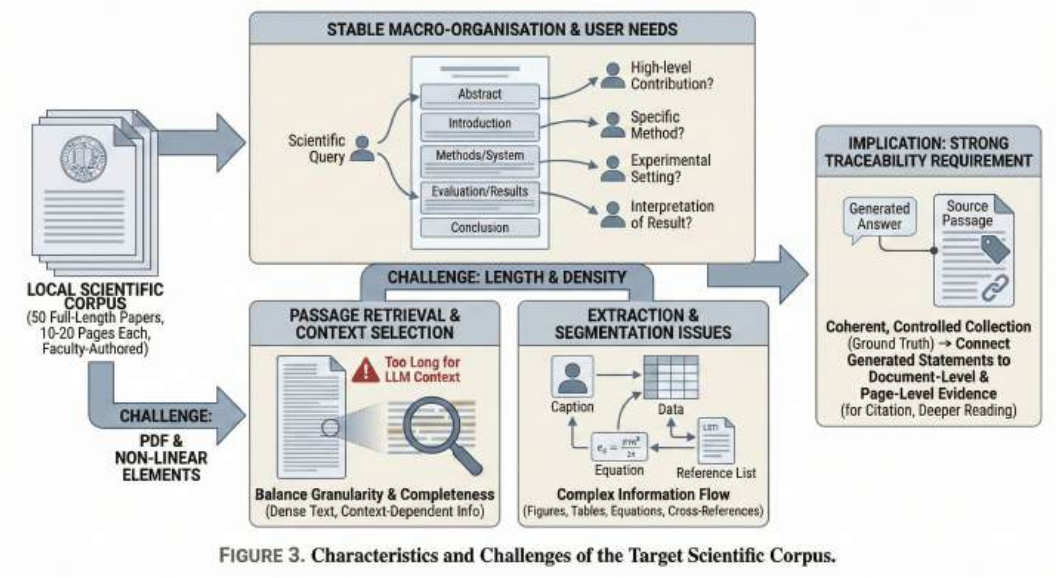

Figure 2.2: Common sources of complexity in scientific PDFs that impact extraction and RAG preprocessing.

controlled collection of documents, enabling systematic evaluation with internal benchmarks and domain-specific questions. Second, it strengthens the requirement of traceability: users may be interested not only in the content of an answer but also in the ability to identify which paper and which part of the paper supports it, for purposes such as citation, discussion in coursework, or deeper reading. For these reasons, the corpus is treated as the primary ground truth source in the system, and the ability to connect generated statements to document-level and page-level evidence is considered a central requirement in the problem formulation.

2.4 Users and Query Scenarios

The system studied in this thesis is designed to support information-seeking and study activities within a university research environment, where the primary knowledge base is a local collection of faculty-authored scientific papers. In this context, the benefit of a question-answering tool lies in its ability to accelerate access to relevant evidence while preserving the scholarly workflow of verification. Rather than replacing the reading of papers, the goal is to reduce the time required

to locate definitions, methodological details, experimental settings, statements of contributions, and to provide a traceable entry point into the original documents.

The first users group is composed by students, including both undergraduate and graduate students, who may use the paper collection to prepare coursework, seminars, and exams, or to gain familiarity with the research topics of the group. Their queries often focus on providing definitions for terms, summaries of contributions, and explanations of methodological choices. In such cases, a system that provides concise answers followed by explicit citations can reduce the effort of navigating multiple documents and enable more focused reading. At the same time, students represent a high-risk group with respect to reliability, because they may not have enough background to recognise subtle inaccuracies; therefore, mechanisms that encourage verification and abstention in the absence of evidence are essential.

The second users group includes researchers, such as PhD candidates and postdoctoral researchers, who use the paper collection for rapid recall of details, comparison across papers, and identification of specific technical components. Their information needs are typically more targeted and may involve fine-grained questions about experimental setups, datasets, hyperparameters, ablation studies, or limitations discussed by the authors. For these users, the ability to retrieve the exact passages supporting an answer is particularly important, both to confirm correctness and to speed up the integration of information into their own research workflow. Researchers also benefit from systematic exploration of the corpus, for instance when scanning related work sections or comparing how different papers define and operationalise a concept.

The third users group is represented by faculty members, who may use the system as a lightweight interface to their own publication set, for tasks such as quickly locating where a concept is introduced, checking how a result is phrased in the final version of a paper, or supporting teaching activities by retrieving passages suitable for discussion. While faculty members typically have the expertise to judge the plausibility of an answer, they still benefit from the time savings of direct passage retrieval and from the transparency provided by citations, especially when the query spans multiple documents.

Across these users, typical usage scenarios can be grouped into a small set of

recurring interaction patterns. One common scenario is definition seeking, where the user asks for the definition of a term or concept as used in a specific paper or within the corpus, often requiring precise wording and references. Another scenario is method-oriented questioning, where the user aims to understand the approach proposed in a paper, including key components, assumptions, and design rationale. A further scenario concerns evidence seeking, in which the user asks for reported results, experimental configurations, or limitations, and expects the system to point to the relevant section of the paper. Finally, comparative scenarios arise when a user asks to contrast two papers or to identify how different works relate to each other; although these queries may require synthesis, they still demand that the synthesis be anchored to identifiable sources(Figure 2.3).

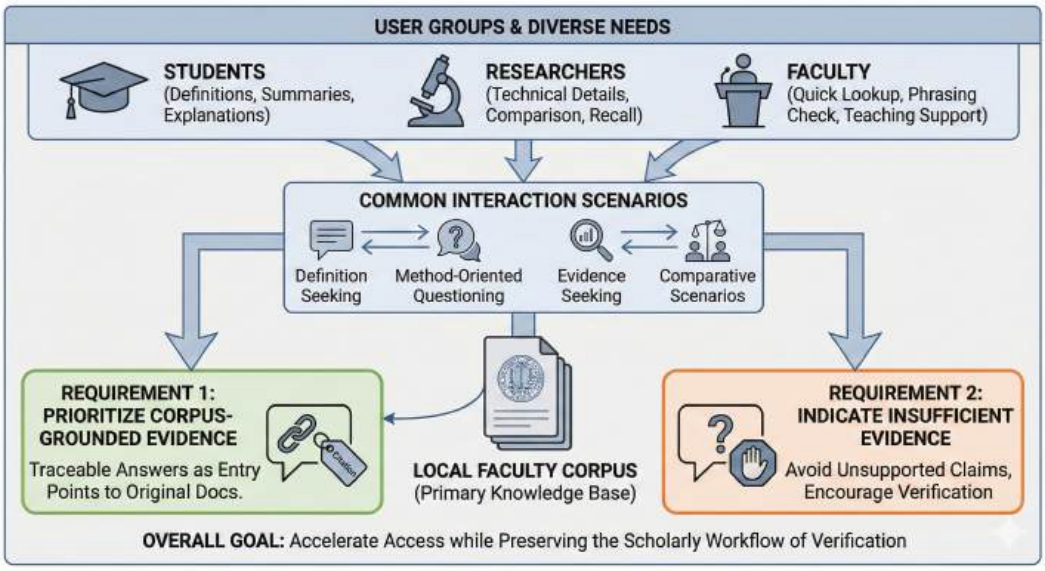

Figure 2.3: Common sources of complexity in scientific PDFs that impact extraction and RAG preprocessing.

These scenarios also make explicit the boundaries of the intended use. The system is not designed to produce authoritative claims beyond what is supported by the local corpus. Instead, it aims to assist users in discovering and validating information contained in the faculty-authored papers. This boundary motivates two behavioural requirements that are central to the case study. First, the system should prioritise corpus-grounded evidence when producing answers, and it should

treat the local papers as the primary source of truth, especially when operating in corpus-only settings or when the query asks for paper-specific details. Second, the system should explicitly indicate when the corpus does not provide sufficient evidence to answer a question, avoiding unsupported statements and encouraging verification against the original documents. In the remainder of this chapter, these needs are translated into functional and non-functional requirements that guide the design and evaluation of the proposed solution.

2.5 Functional Requirements and Expected System Behaviour

This section defines the functional requirements of the proposed system, derived from the academic use case introduced in the previous sections. The central objective is to support question answering over a local corpus of faculty-authored scientific papers, while preserving traceability and enabling users to verify the underlying evidence.

The system must accept natural language queries and return a coherent answer that is grounded in the local document collection. Grounding is operationalised by requiring that the answer be produced with explicit support from retrieved passages of the corpus. Accordingly, the system must implement a retrieval component capable of selecting a small set of passages relevant to the user query, and a generation component that conditions its output on such passages. At the level of functional requirements, the system is expected to behave as a document-grounded question answering tool, treating the local paper corpus as the primary source of truth for paper-specific questions.

A key functional requirement concerns source attribution. For each answer, the system must provide citations that identify the document(s) supporting the main statements of the response. Whenever page-level information is available, citations should additionally report the corresponding page reference, reflecting a *strict* traceability preference at page granularity. When page-level attribution cannot be produced (e.g., due to limitations in PDF text extraction or metadata alignment), the system may still provide document-level citations. Conversely,

when no supporting document can be identified for the query, page-level attribution is not possible, and the system is expected to avoid unsupported claims and to handle the case as insufficient evidence (see the requirement on abstention below).

Another functional requirement concerns handling of insufficient evidence. The system must be able to recognise when the retrieved evidence does not support a definitive answer. This includes scenarios where the query is out of scope with respect to the corpus, where the question is underspecified, or where the available passages do not contain the necessary information. In these cases, the system should avoid producing unsupported claims and should instead provide an abstention-style response that communicates the limitation and encourages verification against the original documents.

The system must support both interactive usage and systematic experimentation. Interactive usage refers to a mode in which a user can pose questions and obtain answers with citations in real time, enabling exploration of the corpus. Systematic experimentation refers to a batch evaluation mode in which the system can be executed over a benchmark dataset of questions, producing structured outputs suitable for comparison across configurations. This dual requirement reflects the engineering nature of the project: the system is designed both as a usable tool and as an experimental platform for studying the impact of design choices on answer quality and traceability.

Finally, the system has to expose configurability of the main pipeline parameters that influence behaviour. In particular, it must be possible to vary retrieval parameters such as the number of retrieved passages, to select different prompt templates used to instruct the model, and to enable alternative operating configurations that control how strictly the answer should adhere to corpus-grounded evidence. This configurability is required to support the experimental questions posed by this thesis and to enable reproducible comparisons across settings.

2.6 Non-functional Requirements and Engineering Constraints

In addition to the functional behaviour described in the previous section, the proposed system is subject to a set of non-functional requirements and practical constraints that derive from the academic setting and from the engineering nature of the project. These requirements guide the design choices discussed in Chapter 4 and shape the evaluation methodology in Chapter 5.

A primary non-functional requirement is reliability in the presence of uncertainty. In this case study, reliability does not simply mean returning an answer for every input; rather, it means avoiding unsupported claims when documentary evidence is insufficient. This requirement is closely tied to traceability, since a response that cannot be linked to supporting sources cannot be verified and is therefore of limited value in scholarly workflows. As a consequence, the system is expected to favour evidence-based answers and to behave conservatively when the retrieval stage does not provide adequate support.

Traceability itself is a non-functional requirement that directly affects user trust and usability. The requirement is not limited to naming the supporting paper, but includes a preference for page-level attribution whenever available. Page-level references enable rapid navigation inside long PDF documents and reduce the cognitive effort required to validate the answer. At the same time, page attribution depends on the quality of PDF processing and metadata alignment, and therefore the system must operate under the constraint that page information may not always be extractable with high confidence for every passage. This motivates a design where document-level citations remain possible even when page-level details are missing, while still preserving a strict preference for page references whenever they can be produced.

Reproducibility is another key requirement, because the thesis aims to compare alternative configurations of the pipeline in a controlled manner. The system must therefore support stable execution settings and structured logging of experimental outcomes. In particular, controlling sources of nondeterminism where possible (e.g., by fixing a seed when supported) and exporting results in a machine-readable format enables consistent comparisons across runs. This requirement is aligned

with the evaluation workflow, where multiple combinations of system configurations are tested and their outputs are analysed quantitatively and qualitatively.

Portability and ease of deployment represent an additional constraint, especially in academic environments where experiments may be reproduced on different machines. For this reason, the system should be runnable through a standardised and isolated environment, limiting the impact of differences in local dependencies and configurations. Container-based deployment supports this objective by enabling a consistent runtime setup for the main components of the pipeline, including the vector database and the LLM backend. This portability requirement also facilitates future maintenance and extension of the system beyond the scope of this thesis.

A further constraint concerns the trade-off between local execution and cloud-based execution. The case study motivates support for a local LLM backend, which can reduce dependence on external services and can provide stronger control over costs and data flows. At the same time, the system should allow optional use of cloud-based models when available, which may offer higher quality or different capabilities. This duality introduces constraints related to credentials management, cost-awareness, and consistency of experimental comparisons across backends. As a result, the system should make the selected backend and configuration explicit at run time, so that results can be correctly interpreted and replicated.

Overall, these non-functional requirements emphasise that the system is designed not only to generate answers, but to do so in a way that supports verification, controlled experimentation, and practical deployment in an academic setting. The next section formalises the problem in more explicit terms by defining inputs, outputs, and the evaluation dimensions implied by the case study.

2.7 Problem Definition and Evaluation Dimensions

This section describes the problem addressed in the case study by outlining the system inputs and outputs, together with the dimensions used to evaluate performance. The description is aligned with the functional and non-functional require-

ments introduced earlier, especially the need for evidence-grounded answers and, whenever possible, strict traceability through citations.

The case study considers a finite corpus of scientific documents composed of faculty-authored full papers available in PDF format. In the experimental setting, the corpus includes 50 documents written in English, with a typical length between 10 and 20 pages. Before retrieval, each document undergoes a processing step in which its text is extracted and segmented into smaller retrievable units (chunks). Each chunk is stored together with metadata that identifies its source document and, whenever available, its page information. This metadata is a key enabler for traceability at both document and page level.

Given a user query expressed in natural language, the system is expected to produce two outputs: a natural-language answer and a set of citations that provide evidence for the main claims. Each citation points to the document that supports a statement and, when reliable page information can be provided, it also includes the relevant page reference. This reflects the strict traceability preference adopted in this thesis: page-level citations are the target whenever they can be produced consistently, while document-level citations remain acceptable when page attribution cannot be determined with sufficient reliability.

A central aspect of the problem is answerability. Not all user queries can be answered based on the local corpus. For this reason, the system must handle both answerable and non-answerable cases. A query is considered answerable when the corpus contains enough evidence to justify a response at the required level of specificity. A query is considered non-answerable when the necessary information is missing from the corpus, when retrieval does not return sufficiently reliable evidence, or when the query is underspecified and admits multiple incompatible interpretations. In non-answerable cases, the expected behaviour is to avoid unsupported claims and to return an abstention-style response that explicitly communicates insufficient evidence, rather than generating a plausible but ungrounded answer.

This problem setting leads to multiple evaluation dimensions. The first dimension is answer quality, referring to whether the generated answer correctly addresses the query and remains consistent with the evidence available in the corpus. The second dimension is grounding, that is, whether the answer content

is supported by retrieved and cited passages rather than relying on unsupported extrapolation. The third dimension is citation correctness. At document level, citation correctness requires that cited documents are relevant and genuinely contain evidence for the statements they support. At page level, citation correctness additionally requires that the referenced pages actually include the supporting evidence and make verification efficient for the user. Since page attribution depends on document processing, the evaluation explicitly recognises document-level citations as a valid outcome when page information cannot be produced, while preserving a strict preference for page-level references whenever available.

Finally, the formulation reflects the experimental perspective of this work. The system is not assessed as a single fixed configuration, but as a family of configurations defined by design choices such as the segmentation strategy, retrieval parameters (for example the number of retrieved passages), the prompt templates used for generation, and operating modes that control how strictly the model should adhere to corpus-grounded evidence. The goal is to study how these choices affect the evaluation dimensions above, with particular attention to robustness under ambiguous and non-answerable queries, and to the traceability requirements typical of academic usage.

Chapter 3

State of the art

3.1 RAG-based architectures for document-grounded question answering

A common solution to question answering over domain-specific document collections is to augment a language model with an explicit retrieval component, rather than relying solely on the model's internal (parametric) knowledge. This approach is generally referred to as RAG and has become a reference architecture for knowledge-intensive tasks in which answers should be grounded in external sources and remain updateable over time. The core idea is to couple a generative model with a non-parametric memory implemented as a searchable index, so that, for each query, relevant passages are retrieved and used as conditioning context for generation. In the original formulation, RAG combines a pretrained sequence-to-sequence model with a dense retriever that accesses a large text corpus, and it explores alternative conditioning schemes, including using the same retrieved set for the entire output or allowing retrieval to vary during generation.[7]

From an engineering perspective, the RAG paradigm addresses two limitations that frequently arise when using LLMs for document-centric question answering. First, it provides a mechanism to incorporate knowledge that is not reliably present in the model training data, which is particularly relevant when the target corpus is local and specialised, such as a collection of academic papers. Second, it enables provenance-oriented workflows, because retrieved passages can be exposed to the

user as supporting evidence, turning the LLM into a natural-language interface to a document repository rather than a standalone oracle.[7] As a result, many systems that aim to support accurate answers over proprietary or curated corpora adopt the same high-level pipeline: document ingestion and indexing, similarity-based retrieval, context construction, and answer generation conditioned on retrieved evidence.

While the general pipeline is shared, related work shows that RAG systems can differ substantially in how they implement and connect retrieval and generation. A first family of approaches treats retrieval as a one-shot step performed prior to generation: the system retrieves the top-k passages for the query, concatenates them into a context window, and prompts the model to answer based on that context. This design is simple and practical, but it makes the final answer highly sensitive to segmentation choices and to the presence of irrelevant or misleading passages in the retrieved set. A second family of approaches investigates tighter integration between retrieval and generation, for instance by iterating or adapting retrieval as generation proceeds, or by introducing additional control stages that critique the draft answer against the retrieved evidence and then revise it accordingly. These variants aim to improve faithfulness and robustness, especially when the initial retrieval is imperfect, but they introduce additional complexity and may require more involved evaluation protocols.

In the context of this thesis, the architectural pattern established by RAG provides a natural reference solution for the target problem: answering queries over a bounded corpus of scientific papers while enabling verification through citations. Consequently, the remainder of this chapter reviews related work by decomposing RAG into its main building blocks: document processing and chunking, dense retrieval and ranking, grounded generation via prompting, citation-aware answering, and evaluation methodologies; and by discussing known failure modes that motivate systematic experimentation in later chapters.

3.2 Building blocks: dense retrieval and vector search for RAG

A central building block in modern RAG pipelines is the retrieval layer, whose role is to select a small set of passages that are relevant to a given query. In many recent systems, retrieval is implemented as *dense retrieval*, where both queries and candidate passages are mapped into a shared vector space and relevance is approximated by a similarity function (e.g., inner product or cosine similarity). Dense retrieval emerged as a practical alternative to purely lexical matching because it can capture semantic similarity beyond exact term overlap, which is particularly important for technical texts where equivalent concepts may be expressed with different terminology. A canonical reference in this line of work is Dense Passage Retrieval (DPR), which introduced a dual-encoder architecture trained to retrieve Wikipedia passages for open-domain question answering by maximising similarity between question and supporting passage embeddings.[8]

From an engineering standpoint, dense retrieval decomposes into two phases. During indexing, the document collection is segmented into retrievable units and each unit is embedded by a passage encoder, producing a set of vectors that can be stored in an index. At query time, the user question is embedded by a query encoder, and nearest-neighbour search is performed to retrieve the top-k passage vectors. This design is compatible with scalable vector search infrastructures and allows retrieval to be implemented efficiently over large corpora.[8] In a RAG system, the retrieved passages form the evidence base that is injected into the model prompt, making retrieval quality and ranking highly influential factors for the final answer.

A practical consequence of dense retrieval is that the system must also select an indexing and search strategy that supports efficient similarity queries. Vector databases and approximate nearest neighbour (ANN) indexes are commonly used to enable low-latency top-k search in high-dimensional embedding spaces. While the exact indexing method and implementation vary across systems, the underlying principle is shared: retrieval returns a small evidence set that should be both relevant and sufficiently diverse to cover the information needed to answer the query. In document-grounded question answering, this requirement introduces a

design trade-off. If k is too small, critical evidence may be missed and the generator may either abstain or fill gaps with unsupported content. If k is too large, irrelevant or weakly related passages may be included, increasing noise and potentially reducing faithfulness because the model may attend to misleading context rather than to the correct evidence.

Related work also highlights that retrieval quality alone is not sufficient to guarantee grounded answers, but it remains a necessary condition. When retrieval fails to include the relevant passage among the top-k results, generation is inherently constrained and any detailed answer becomes risky. Conversely, even when retrieval succeeds, ranking and context construction decisions shape what the generator actually sees, especially under context-length limits. Therefore, many systems complement dense retrieval with additional selection or ranking mechanisms (e.g., reranking models or heuristic filters) to prioritise the most supportive evidence and to reduce the probability of injecting distracting context into the prompt.

In the setting of this thesis, dense retrieval provides a natural solution for searching a local corpus of academic papers, where queries often require semantic matching over specialised terminology. The retrieval layer defines the interface between the indexed paper collection and the generative model, and it is a key source of controllable parameters for experimental analysis, including the choice of embedding model, the vector indexing backend, and retrieval hyperparameters such as top-k. These aspects will be implemented in the proposed system in Chapter 4 and evaluated systematically in Chapter 5.

3.3 Document processing and chunking for scientific PDF

A recurring practical challenge in document-grounded question answering is that source documents are typically longer than the amount of context that can be provided to a language model at inference time. As a result, related work on RAG systems commonly adopts a two-step strategy: documents are first converted into a representation suitable for retrieval, and the retrieval layer then selects a small

subset of this representation to condition generation. A key implication is that the *retrievable unit* becomes a design choice: how documents are segmented directly affects both retrieval effectiveness and end-to-end answer quality.[9, 10]

A standard solution is to transform each paper into a collection of retrievable units through segmentation, often referred to as *chunking*. Chunking enables retrieval at a granularity finer than the full document, allowing the system to return localised passages that likely contain the evidence needed to answer a query. Empirical studies show that segmentation strategy can have a measurable impact on downstream performance: overly coarse chunks reduce precision and can dilute relevance, while overly fine chunks can fragment evidence and harm both retrieval and generation.[9, 11]. Figure 3.1 illustrates common chunking strategies and the practical trade-offs between context preservation and retrieval precision.

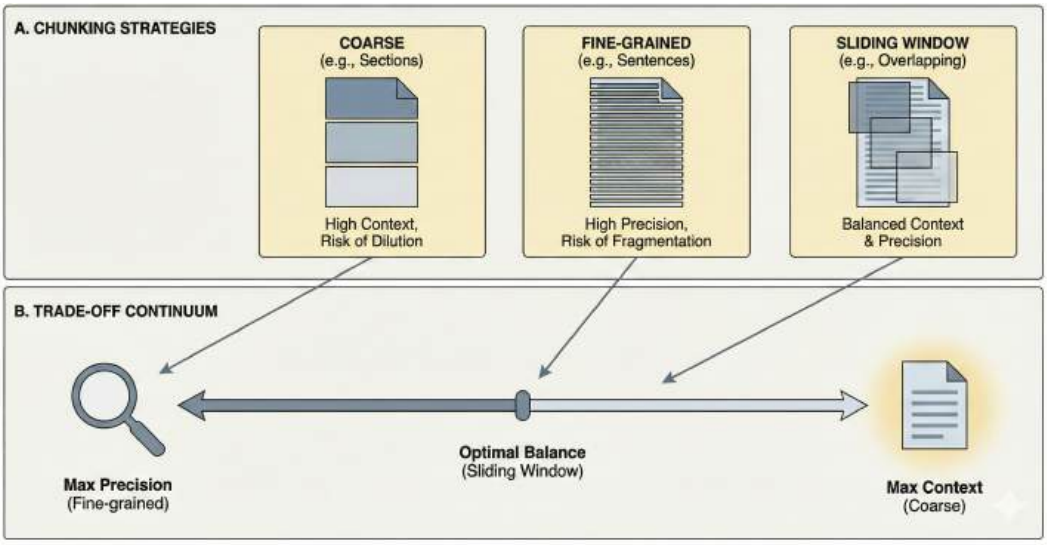

Figure 3.1: Chunking stratedgies and trade offs.

In practice, related work explores both structure-aware and structure-agnostic chunking strategies. Structure-aware methods attempt to preserve boundaries induced by document organisation, such as section headings or paragraphs, and are motivated by the observation that scientific writing is organised into semantically coherent blocks (e.g., methods, evaluation, limitations). Structure-agnostic strategies instead rely on simpler heuristics such as fixed token windows, sometimes with overlap, and trade structural fidelity for ease of implementation and predictable

chunk size.[11] Recent work also investigates *adaptive* approaches that select different granularities depending on the query or route between multiple representations, aiming to balance context completeness and retrieval precision.[10]

The PDF format introduces additional variability. PDF documents are optimised for visual consumption and may contain multi-column layouts, figures, tables, and equations. Depending on the extraction pipeline, these elements can be ignored, partially captured, or converted into noisy text, which in turn affects how well chunks preserve coherent evidence for retrieval.[9] For this reason, many RAG systems operating over PDF corpora make explicit assumptions about which content types are in scope for evidence-grounded question answering and design their chunking and metadata strategies accordingly.

Chunking choices are also tightly coupled with traceability. In academic settings, retrieved evidence must be connectable to the original paper; therefore, related work commonly stores metadata alongside each chunk, including document identifiers and, when possible, location information such as page numbers. This metadata supports citation-aware generation and verification workflows, and it motivates evaluating not only answer quality but also citation correctness at document and page granularity.

Overall, the state of the art treats document processing and segmentation as first-class components of a RAG pipeline. They determine the unit of retrieval, affect the noise-to-signal ratio of retrieved contexts, and influence whether evidence can be cited precisely. Consequently, segmentation strategy is commonly treated as a controllable design parameter and studied in combination with retrieval hyperparameters such as top-k and prompting strategies that govern how the generator consumes the retrieved context.[9, 11]

3.4 Generation layer: prompting strategies for grounded answering

In RAG systems, the generation layer does not merely "summarise" retrieved passages: it must transform retrieved evidence into an answer that is both useful and faithful to the supporting documents. Related work consistently treats *prompt-*

ing as a key control surface for this behaviour, because the prompt specifies (i) what the model should answer, (ii) which evidence it should use, and (iii) which constraints it must respect (e.g., cite sources, avoid speculation, abstain when evidence is missing). In practice, prompt engineering has evolved into a structured set of techniques and best practices, and systematic surveys have proposed taxonomies and vocabularies that clarify how prompts can be designed to steer model behaviour[12]. In this context, structured frameworks such as COSTAR-A provide a systematic way to enhance LLM performance on point-of-view questions [13]. Furthermore, recent evidence suggests that techniques like prompt repetition can materially improve the performance of non-reasoning LLMs by reinforcing key constraints during generation [14].

A common prompting pattern for grounded question answering is to explicitly instruct the model to rely on the provided context and to avoid introducing information that is not supported by the retrieved passages. This is often combined with formatting constraints, for example requiring a short, direct answer followed by a citation block, or requiring that each claim be associated with a source reference. Such constraints are particularly relevant in academic settings, where the usefulness of an answer depends on whether users can verify the supporting evidence in the underlying papers. Prompting is also used to encourage conservative behaviour under uncertainty. For instance, prompts may instruct the model to respond with a refusal or an "insufficient evidence" statement when the retrieved context does not contain the information needed to answer the query.[12]

However, related work indicates that abstention and conservative behaviour are not guaranteed by instruction alone. Empirical studies on scientific question answering show that models may fail to abstain appropriately under context perturbations and may still produce confident but unsupported responses, especially when the prompt or context provides superficial cues.[15] This motivates treating abstention not as a cosmetic response option, but as a measurable property that must be validated experimentally. In RAG pipelines, this validation is particularly important because uncertainty may arise either from the absence of evidence in the corpus or from imperfect retrieval that fails to surface the relevant passage among the retrieved results.

Beyond single-shot prompts, some lines of work introduce *structured prompting*

and iterative prompting loops to improve faithfulness to evidence. A representative direction reframes RAG as *retrieval-augmented reasoning* and addresses the risk that the model's reasoning trajectory diverges from the constraints imposed by retrieved passages. These approaches use critique-and-revision steps at inference time to detect and correct evidence misalignment, effectively adding an explicit control layer on top of the base prompt.[6] While such methods increase complexity, they highlight an important point for system design: the prompt is not only a natural-language instruction, but a mechanism that can be combined with structured intermediate steps to enforce grounding requirements.

Overall, related work supports two conclusions that are directly relevant to the case study of this thesis. First, prompting is a first-class design variable in RAG pipelines, because it governs how retrieved evidence is consumed and how strictly the model adheres to traceability constraints.[12] Second, prompting choices should be evaluated empirically, especially for behaviours such as abstention and citation discipline that are critical in academic usage.[15] These considerations motivate the experimental comparison of alternative prompt templates in later chapters, where prompt design is treated as a controllable parameter alongside retrieval settings.

3.5 Citation-aware generation and traceability

A distinctive requirement in academic question answering is that answers must be *checkable*. In practice, this implies that generated responses should be accompanied by citations that allow a reader to locate supporting evidence in the original documents. In RAG pipelines, citations are naturally tied to retrieval: since the system already selects passages from the corpus, it can expose the provenance of those passages and use it to justify the answer. However, related work shows that citation support is not an automatic by-product of retrieval. Producing *accurate* citations is a separate challenge because the generator may (i) attribute a claim to an incorrect source, (ii) provide a correct document but an incorrect location, or (iii) introduce citations that do not actually support the stated content. Figure 3.2 shows an example of citation-aware generation, where the model is guided to connect claims to explicit evidence.

Recent work in LLM-assisted academic writing explicitly treats citation ac-

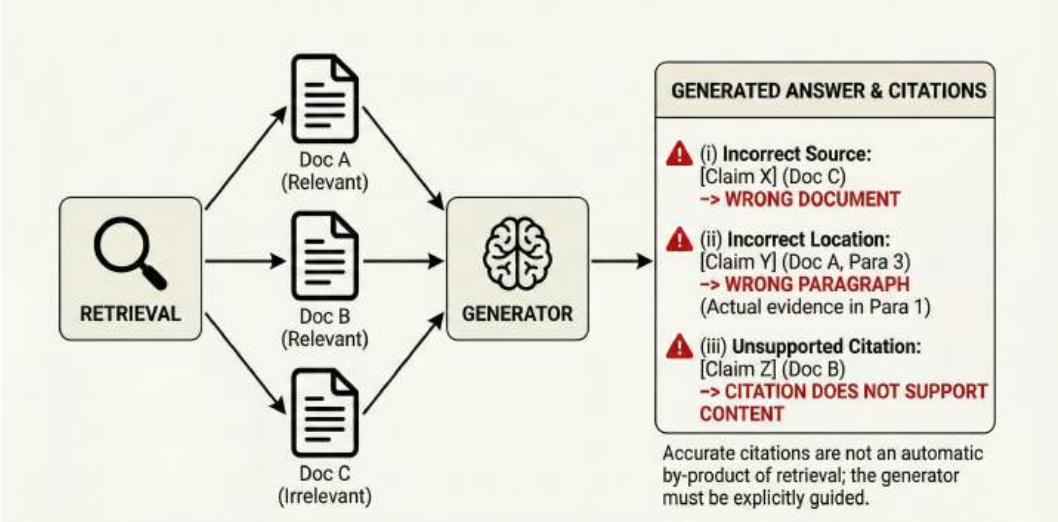

Figure 3.2: Citation-aware RAG design for scientific question answering.

curacy as a first-class objective. For example, ScholarCopilot proposes training strategies for generating academic text with accurate citations, emphasising that the usefulness of an assistant in scholarly workflows depends not only on linguistic quality but also on the correctness of source attributions.[4] This perspective aligns with the case study of this thesis, where citations serve both as a trust mechanism and as a navigation aid to accelerate paper reading. In particular, citations are most useful when they support efficient verification, which motivates page-level traceability whenever it can be obtained reliably.

From a system-design standpoint, citation-aware RAG requires two conditions. First, the document processing pipeline must preserve provenance metadata for each retrievable unit, including at least a stable document identifier and, when feasible, location information such as page numbers or section labels. Second, the generation process must be constrained to reference only the sources that are available through retrieval, rather than freely producing bibliographic-looking references. In practice, this constraint is often implemented through prompt-level instructions that require citations to be drawn from the retrieved context and through output schemas that force the model to attach citations to the answer in a structured way. Prompt engineering surveys describe citation enforcement as an instance of output control and grounded generation constraints.[12]

Despite these mechanisms, citation behaviour remains fragile under imperfect retrieval and under context perturbations. When the retrieved set contains partially relevant evidence, the model may overgeneralise and attach citations that are only loosely related to the claim. When retrieval returns misleading passages, the model may faithfully cite those passages while still producing an incorrect answer, which highlights that "being cited" is not equivalent to being correct. Robustness studies on misleading retrievals emphasise that RAG systems can be systematically diverted by unhelpful context, motivating evaluation protocols that measure not only answer correctness but also whether citations are actually supportive.[1] Complementary work on retrieval-aware reasoning further shows that misalignment can occur even when evidence is present, because the generation process may fail to integrate the retrieved information coherently; in such cases, citations alone do not guarantee grounded reasoning.[6]

For these reasons, related work increasingly distinguishes between multiple levels of citation quality. At a minimum, *document-level correctness* requires that cited papers are relevant and contain evidence supporting the main claims. A stricter notion is *location-level correctness*, which requires that the cited location (e.g., a page reference) actually includes the supporting evidence and enables efficient verification. This distinction is especially important for scientific PDFs, where long documents and dense content make page-level navigation valuable, but where page attribution can be affected by extraction and alignment errors. Consequently, systems that aim to support scholarly workflows typically adopt a conservative stance: they should prefer precise citations when available, avoid fabricating locations, and treat citation correctness as an evaluation target rather than an assumed property.

In summary, related work motivates treating citation-aware generation as a dedicated component of a RAG system rather than as a superficial formatting feature. Accurate citations improve trust and usability, but they require both provenance-preserving document processing and generation-time constraints that tie claims to retrieved evidence.[4, 12] These considerations directly inform the design and evaluation choices in this thesis, where traceability is treated as a core requirement and page-level citation accuracy is studied whenever the underlying metadata allows it.

3.6 Evaluation methodologies for RAG systems

Evaluating RAG systems is more complex than evaluating either retrieval or generation in isolation, because the final behaviour emerges from their interaction. Related work therefore proposes evaluation methodologies that decompose performance into multiple dimensions, reflecting the end-to-end objectives of evidence-grounded question answering. In the context of academic document QA, these dimensions typically include the quality of retrieved evidence, the correctness and usefulness of the generated answer, and the degree to which the answer is grounded in and traceable to the supporting sources.

A first family of approaches evaluates the *retrieval* component using information-retrieval metrics, such as recall at k, precision at k, and ranking-based measures (e.g., MRR or nDCG), when some notion of relevant passages is available.[16] In document QA settings, constructing passage-level relevance labels is often costly; as a result, many works adopt proxy evaluations, such as assessing whether the retrieved set contains the information necessary to answer the question or whether it includes the correct document among the top-k.[16] These retrieval-centric views are important because retrieval failures place a hard upper bound on grounded answer quality: when the necessary evidence is not retrieved, the system can either abstain or risk generating unsupported claims.

A second family of approaches focuses on *answer quality*. Traditional automatic metrics based on n-gram overlap (e.g., BLEU/ROUGE) are often inadequate for open-ended question answering, especially when multiple correct phrasings exist and answers may be partially correct.[17, 18] Consequently, related work increasingly relies on semantic similarity measures and LLM-based judging protocols that assess correctness and usefulness beyond surface form.[17, 18, 19] In scholarly settings, answer correctness is often assessed together with informativeness and with alignment to the terminology and claims present in the source documents, motivating evaluation criteria that explicitly consider factual consistency with provided evidence.[18, 19]

This motivates a third evaluation dimension: *groundedness* (or faithfulness), which aims to measure whether the answer is supported by the retrieved evidence. In recent work, groundedness is treated as a separate target and is assessed

by checking whether the claims in the answer can be inferred from the provided context. RAGAs is an example of an automated evaluation framework designed specifically for RAG pipelines: it proposes reference-free metrics that score faithfulness, answer relevance, context relevance, and context recall by using model-based grading prompts(Figure 3.3), providing a structured way to evaluate both retrieval and generation behaviours without requiring gold answers for every query.[16]

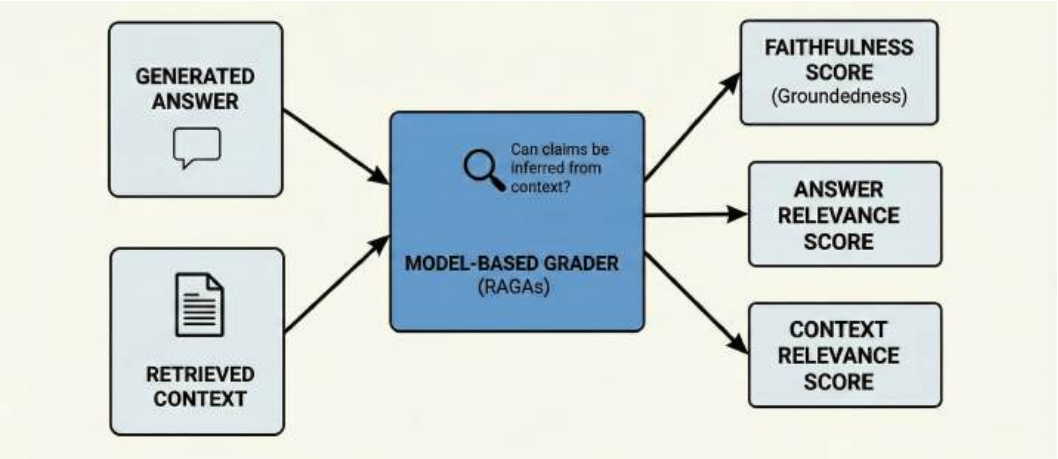

Figure 3.3: Evaluating Groundness.

In academic corpora, the evaluation space is further enriched by *citation quality*. Since the system is expected to provide citations that support verification, citation correctness becomes measurable at multiple granularities. At document level, evaluation checks whether the cited paper actually contains supporting evidence. At location level, evaluation checks whether the cited location (e.g., page reference) enables the reader to find the supporting passage efficiently. Related work on citation-aware generation highlights that citation accuracy should be treated as an explicit objective and a measurable outcome, because models may otherwise provide plausible but incorrect attributions.[4] In addition, robustness studies on misleading retrievals show that RAG systems can be systematically diverted by unhelpful contexts, reinforcing the need for evaluation protocols that analyse failure modes beyond average answer quality.[1]

Finally, related work emphasises the importance of evaluation protocols that support *comparative* analysis across system configurations. Because RAG pipelines expose multiple controllable parameters (including chunking strategy, retrieval

depth (k), prompt templates, and LLM backends) a meaningful evaluation should allow systematic variation of these factors and consistent reporting of results. This motivates structured logging and machine-readable output formats to enable aggregation, comparison, and error analysis across experimental runs. In this thesis, the evaluation methodology follows this line by treating configuration choices as experimental variables and by measuring not only answer quality but also traceability and citation correctness.

Overall, the state of the art motivates evaluation methodologies that treat RAG as a multi-objective system: retrieval performance constrains what can be answered, answer quality captures usefulness and correctness, groundedness captures evidence support, and citation metrics capture traceability. These dimensions provide the conceptual basis for the experimental protocol described in Chapter 5.

3.7 Failure modes and robustness challenges

Although RAG has become a reference solution for document-grounded question answering, related work shows that retrieval augmentation does not automatically guarantee better answers. Instead, RAG pipelines exhibit characteristic failure modes that arise from the interaction between retrieval, context construction, and generation. Understanding these limitations is essential for designing evaluation protocols and for motivating the systematic comparison of configurations in later chapters.

A first major challenge concerns *robustness to imperfect retrieval*. In realistic settings, retrieval can return irrelevant, weakly related, or even misleading passages. Robustness studies demonstrate that RAG systems can underperform a zero-shot baseline when retrieved evidence is misleading, because the generator tends to follow the provided context even when it contradicts the correct answer.[1] This observation highlights that retrieval can introduce an additional attack surface: if the evidence set is noisy or adversarial, the generator may produce confidently grounded-looking answers that are nonetheless incorrect. Consequently, robustness must be treated as an empirical property, requiring specific investigation into the resilience of RAG methods across stratified information-seeking tasks [20].

A second challenge is *retrieval-aware reasoning misalignment*. Even when relevant evidence is retrieved, the model may fail to integrate it into the reasoning process in a coherent and faithful way. "Retrieval is Not Enough" shows that the presence of supporting context does not guarantee correct reasoning, motivating test-time critique and optimisation mechanisms that explicitly enforce alignment between intermediate reasoning and retrieved evidence.[6] This line of work suggests that the weak link in RAG pipelines is not only retrieval quality, but also the generator's ability to use evidence correctly, especially for multi-step or compositional questions.

A third challenge relates to the *use of long contexts*. As RAG systems retrieve multiple passages, the resulting prompt can become long, increasing noise and making it harder for the model to locate and exploit the relevant evidence. Recent empirical results show that increasing context length can harm performance even under favourable retrieval conditions, indicating that models may struggle to consistently attend to the right information as the context grows.[2] This issue is particularly relevant for scientific papers, where answers may depend on precise details and where retrieved evidence can easily include dense but only partially relevant passages.

Finally, the above limitations interact with the traceability requirements of academic usage. Citations can increase user trust and enable verification, but related work emphasises that citation presence does not guarantee correctness. A system may cite the retrieved evidence faithfully while still producing an incorrect answer if the evidence itself is misleading or if the reasoning step is misaligned. This further motivates evaluation methodologies that jointly analyse answer correctness, groundedness, and citation correctness rather than treating citations as a cosmetic output feature.[4, 1]

Overall, related work indicates that building a RAG system for academic papers requires not only implementing a retrieval-and-generation pipeline, but also selecting design parameters and prompting strategies that mitigate these failure modes. These findings justify the experimental focus of this thesis on robustness under ambiguous and non-answerable queries, on the impact of retrieval depth and chunking, and on the role of prompt templates in enforcing conservative, evidence-based behaviour.

Chapter 4

Proposed Solution

4.1 System architecture and main components

This chapter describes the design and implementation of the proposed system for paper-grounded question answering over a local corpus of faculty-authored scientific PDFs. The system follows a Retrieval-Augmented Generation (RAG) architecture, in which a retrieval layer selects evidence from the indexed document collection and a generation layer produces an answer conditioned on that evidence. The implementation is designed to support both interactive usage and systematic experimentation across multiple configurations, with particular emphasis on traceability through citations and reproducibility through containerised deployment and structured logging.

At a high level, the pipeline consists of six stages: PDF ingestion and preprocessing, chunk creation, embedding computation and vector indexing, similarity-based retrieval, prompt-based answer generation, and evaluation with structured output export. Figure 4.1 provides an overview of the end-to-end workflow and highlights the interfaces between the main components.

The two operational flows are:

- **Offline (indexing):** PDF corpus → text extraction → chunking → embedding → Qdrant storage.
- **Online (query):** User question → embedding → similarity search → con-

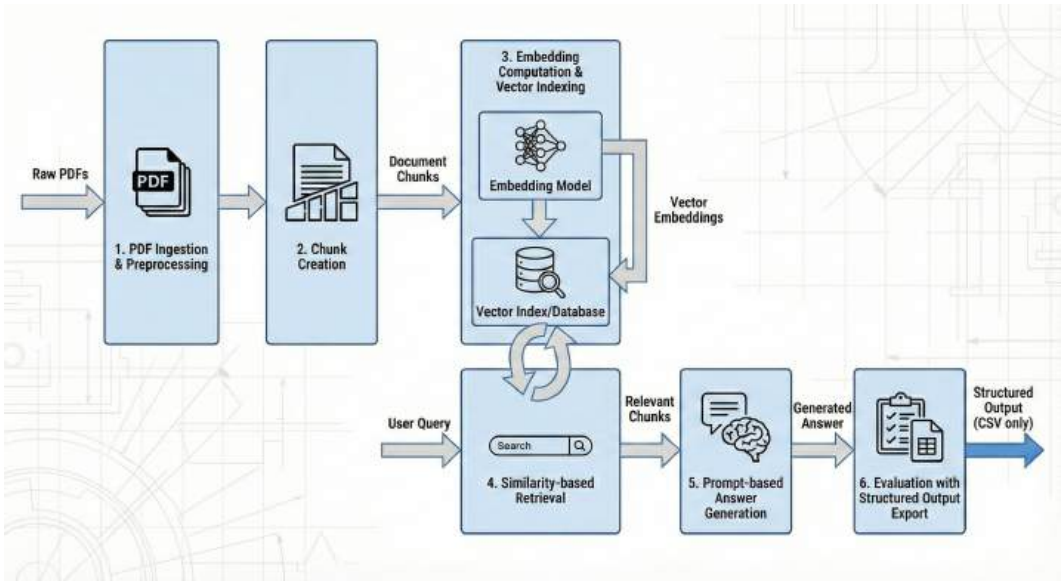

Figure 4.1: High-level architecture of the proposed RAG system generation.

text construction → LLM generation → answer with citations. This is triggered either interactively or through the evaluation pipeline.

A key design decision is the separation between retrieval and generation: the retrieval stage is deterministic and model-agnostic (it depends only on the embedding model and the Qdrant index), while the generation stage is parameterised by the choice of LLM backend and prompt template. This separation enables controlled experiments where the same retrieved context is processed by different models, isolating the effect of the generation component on answer quality.

More specifically, the architecture follows the standard RAG paradigm and is decomposed into five sequential stages, each implemented as an independent Python module:

- **PDF Preprocessing** — Extracts raw text from each page of the input PDF documents, preserving page-level provenance metadata.
- **Chunking and Indexing** — Splits the extracted text into overlapping chunks of fixed size, encodes each chunk into a dense vector using a Sentence Transformer model, and stores both the vector and the associated metadata (source filename, page number, text) in a Qdrant vector database.

- **Retrieval** — At query time, encodes the user question with the same embedding model and performs a cosine-similarity search against the Qdrant collection, returning the top-k most relevant chunks.
- **Generation** — Formats the retrieved chunks into a structured context prompt, selects one of five configurable prompt templates, and sends the prompt to the LLM backend (either a local Ollama instance or the Azure OpenAI API) to produce a grounded answer with inline references.
- **Evaluation** — Compares the system output against a benchmark dataset of gold-standard answers, computing three metrics: source accuracy, page accuracy, and answer similarity.

The entire system is containerised using Docker Compose, which orchestrates three services: Qdrant (vector database, exposed on port 6333), Ollama (local LLM inference server, port 11434), and the RAG application itself. This setup ensures full reproducibility: a single `docker compose up` command provisions all dependencies, and evaluation experiments can be replayed identically via command-line parameters including model name, prompt template identifier, seed, and operating mode.

4.2 PDF ingestion and preprocessing

The first stage of the pipeline ingests each scientific paper in PDF format and converts it into a textual representation suitable for indexing and retrieval. The goal of preprocessing is twofold. First, it produces a text stream that can be segmented into retrievable units in later stages. Second, it preserves provenance information that enables traceability, so that each derived unit can be linked back to its source document and, when feasible, to a page-level location. The corpus is managed as a directory of PDF files. During ingestion, the pipeline enumerates available documents and builds the list of inputs to be processed. The corpus is managed as a directory of PDF files. During ingestion, the pipeline verifies that the configured path exists, enumerates all files in the directory, and retains only those with a `.pdf` extension. The resulting list of document paths defines a fixed

local snapshot of the corpus and is used as input for the subsequent extraction and indexing steps.

PDF is a layout-oriented format, and text extraction may introduce artefacts such as spurious line breaks, hyphenation across lines, and loss of structural cues. To support traceability, the pipeline extracts text on a page-by-page basis and retains the page number alongside the extracted content. The pipeline performs page-level text extraction to preserve provenance metadata. Only non-empty pages are retained, each paired with its page index. This page information is propagated to downstream stages to enable page-level citations. As shown in Algorithm 1, page-level extraction preserves provenance information that later enables page-referenced citations.

Algorithm 1: Offline ingestion and page-level text extraction

Input: PDF corpus directory $\mathcal{D}$
Output: For each document: list of page-level text blocks with provenance metadata

```
pdfs ← list_pdf_files(D);
foreach pdf in pdfs do
    pages ← ∅;
    reader ← open_pdf(pdf);
    for p ← 1 to num_pages(reader) do
        txt ← extract_text(reader, p);
        if txt is not empty then
            pages.append({page: p, text: txt});
    store_document(pdf, pages);
```

The output of preprocessing is therefore a sequence of page-level text blocks for each paper, together with stable document identifiers derived from the source file paths. This representation is passed to the chunking stage described in Section 4.3. Preserving page indices at this stage is essential for the traceability requirement introduced in Chapter 3 when an answer is grounded on retrieved chunks derived from page-level blocks, the system can include page references in citations whenever such information is available.

4.3 Chunking strategy and provenance metadata

After preprocessing, each paper is represented as a sequence of page-level text blocks (Section 4.2). The next stage partitions each page into smaller retrievable units (*chunks*) that constitute the atomic elements indexed in the vector database. Chunking is a critical step for document-grounded question answering because it defines the granularity of retrieval. Chunks that are too large tend to reduce retrieval precision and inject unnecessary context into the prompt, whereas chunks that are too small can fragment evidence and remove local context that is required to interpret technical statements.

In the proposed system, chunking is performed on a per-page basis. This choice is primarily motivated by traceability requirements: since each chunk originates from a specific page, the system can propagate page information through indexing and retrieval and provide page-level citations whenever available. The chunking procedure uses a fixed maximum chunk size and a controlled overlap to mitigate boundary effects. The overlap reduces the risk that a relevant statement is split across chunk boundaries and becomes difficult to retrieve or interpret.

Table 4.1 reports the parameters used in the final prototype. The chosen values reflect a trade-off between context completeness and index size. In particular, the overlap is kept smaller than the chunk size to limit redundancy while still preserving continuity across adjacent chunks.

Table 4.1: Main segmentation and retrieval parameters used in the prototype.

Parameter	Value	Role
`CHUNK_SIZE`	1000	Maximum chunk length (characters)
`CHUNK_OVERLAP`	200	Overlap between consecutive chunks
`TOP_K`	5	Number of retrieved chunks per query

The chunking algorithm implements a punctuation-aware splitting heuristic. Starting from a tentative end position defined by `CHUNK_SIZE`, the algorithm shifts the boundary backward until it finds a delimiter (punctuation or newline) so that chunks are more likely to end at natural textual boundaries. If the boundary shift would collapse the chunk length below the overlap threshold, the algorithm

falls back to the original boundary to avoid generating excessively short chunks. Algorithm 2 summarises the punctuation-aware segmentation strategy adopted to balance retrieval focus and local context preservation.

Algorithm 2: Punctuation-aware chunking with overlap

Input: Page-level text blocks $\{(p, txt_p)\}$, max chunk length L, overlap size O
Output: List of chunks with provenance metadata

```
chunks ← ∅;
buffer ← empty string;
buffer_start_page ← undefined;
foreach page block (p, txt_p) in reading order do
    if buffer is empty then
        buffer_start_page ← p
    units ← split_on_punctuation(txt_p);
    foreach u in units do
        if length(buffer + u) ≤ L then
            buffer ← buffer + u;
        else
            chunk ← buffer;
            meta ← {page_start : buffer_start_page, page_end : p};
            chunks.append({text: chunk, meta: meta});
            buffer ← tail(chunk, O) + u;
            buffer_start_page ← p;
if buffer is not empty then
    chunks.append({text: buffer, meta: {page_start: buffer_start_page}});
return chunks;
```

In addition to text segmentation, the chunking stage is responsible for building the provenance metadata required for citation-aware answering. Each chunk is stored with: a stable reference to the source document, the originating page number, a unique chunk identifier, and the chunk text. This metadata schema enables document-level traceability in all cases and page-level traceability whenever page information is available from preprocessing. Each chunk is stored together with a compact payload that preserves provenance information required for traceability. In particular, the payload links the chunk to its source document and, whenever

available, to the originating page range. This metadata is persisted in the vector database alongside the embedding vector and is later used to generate verifiable citations in the final answer.

This representation is later used during indexing and retrieval. In particular, the `source` and `page` fields are propagated to the retrieval output and can be surfaced in the final answer as citations. The choice of page-level chunking, punctuation-aware boundaries, and overlap therefore supports the overall system objectives of evidence grounding and strict traceability whenever page information is available.

4.4 Embeddings and vector indexing

After chunk creation (Section 3), each chunk must be mapped into a dense vector representation in order to support semantic similarity search. The system uses the sentence-transformer model `all-MiniLM-L6-v2` to compute embeddings for both document chunks and user queries. Embeddings are computed offline during ingestion for all chunks in the corpus, then stored in a vector database so that retrieval can be performed efficiently at query time.

4.4.1 Vector representation

Let $f(\cdot)$ denote the embedding function implemented by the chosen sentence-transformer model. For each chunk c, the system computes a vector $\mathbf{v}_c = f(c)$ and stores it as the primary indexable representation of the chunk. This design is consistent with dense-retrieval practice, where relevance is approximated by vector similarity in a shared embedding space. The choice of `all-MiniLM-L6-v2` reflects a trade-off between embedding quality and computational efficiency, as the model can be run locally during ingestion while producing compact representations suitable for similarity search.

4.4.2 Index structure and payload schema

The vector database is implemented with Qdrant, which stores vectors together with an associated payload. In this system, the payload is used to preserve provenance and to enable citation-aware answering: each stored vector is accompanied by fields that identify the source paper, the originating page, and the chunk identifier. In addition, the original chunk text is retained to support prompt construction at query time.

Table 4.2 summarises the payload schema used for indexing. The design follows a simple principle: the vector is used for retrieval, while the payload provides the metadata necessary to reconstruct evidence and citations when a point is retrieved.

Table 4.2: Payload fields stored alongside vectors in the Qdrant index.

Field	Type	Description
`source`	string	Paper identifier (e.g., PDF filename)
`page`	integer	Originating page number (when available)
`chunk_id`	integer	Unique chunk identifier within the ingestion run
`text`	string	Chunk textual content (used for prompt construction)

4.4.3 Index creation and insertion

At ingestion time, the system initialises a Qdrant collection configured for similarity search (Each point in the collection stores both a 384-dimensional vector and a payload containing the chunk's metadata) and inserts one point per chunk. Each point consists of the embedding vector and the payload described above. In the final prototype, the collection is configured with a cosine distance, which is a common choice when embeddings are compared by cosine similarity at retrieval time. In the implementation, The point ID is derived from an MD5 hash of the chunk's source, page, and the first 20 characters of its text. This makes the insertion idempotent: if the ingestion pipeline is executed multiple times on the same corpus, Qdrant's upsert operation will overwrite existing points with identical IDs rather than creating duplicates. This design eliminates the need for a separate deduplication step and makes re-ingestion safe.

Algorithm 3 summarises the indexing procedure that stores chunk embeddings and provenance payloads, as well as the top-k retrieval step used at query time.

Algorithm 3: Vector indexing and retrieval in the vector database

Input: Chunks $\mathcal{C}$ with payload, embedding model E, vector DB V, batch size B, top-k

Output: Index in V and retrieval procedure

Indexing (offline);
batch $\leftarrow \emptyset$;
foreach *chunk* c *in* $\mathcal{C}$ **do**
    vec $\leftarrow Embed(E, c.text)$;
    point $\leftarrow \{id : new\_id(),\ vector : vec,\ payload : c.meta\}$;
    batch.append(point);
    if *size(batch)* $= B$ **then**
        Upsert(V, batch);
        batch $\leftarrow \emptyset$;
if *batch not empty* **then**
    Upsert(V, batch);

Retrieval (online);
Function *Retrieve*(q, k)**:**
    $q_{vec} \leftarrow Embed(E, q)$;
    $R \leftarrow SearchTopK(V, q_{vec}, k)$;
    return R;

This indexing strategy supports the retrieval stage described in Section 4.5. When a query is issued, Qdrant returns the identifiers and payload of the top-ranked points, enabling the system to reconstruct the evidence set (chunk texts) and to surface document-level and page-level citations in the final response whenever page information is available.

4.5 Retrieval and context construction

At inference time, the system answers a user query by first retrieving a small set of evidence chunks from the indexed corpus and then constructing a prompt that injects these chunks as context for generation. Retrieval is implemented as dense

vector search in Qdrant and is controlled by the parameter `TOP_K`, which is set to 5 in the default configuration. This choice reflects a trade-off between evidence coverage and noise: retrieving too few chunks may omit relevant information, while retrieving too many may introduce distracting or misleading context and unnecessarily enlarge the prompt.

4.5.1 Dense retrieval as similarity search

Let $f(\cdot)$ denote the embedding function used by the model `all-MiniLM-L6-v2`. Given a query q, the system computes its embedding $\mathbf{v}_q = f(q) \in \mathbb{R}^d$. Each indexed chunk $c \in \mathcal{C}$ has a precomputed embedding $\mathbf{v}_c = f(c)$. Retrieval can be formalised as selecting the top-k chunks according to a similarity function, for example cosine similarity:

$$\mathrm{sim}(\mathbf{v}_q, \mathbf{v}_c) = \frac{\mathbf{v}_q \cdot \mathbf{v}_c}{\|\mathbf{v}_q\| \, \|\mathbf{v}_c\|}.$$

The retrieved evidence set is then

$$\mathcal{N}_k(q) = \mathrm{TopK}_{c \in \mathcal{C}} \, \mathrm{sim}(\mathbf{v}_q, \mathbf{v}_c),$$

where $k = \texttt{TOP\_K} = 5$ by default.

In the implementation, the query embedding is computed online and passed to Qdrant, which returns the k most similar indexed points together with their payload and a similarity score. At query time, the system performs dense retrieval over the vector index. The user query is embedded with the same embedding model used during indexing and matched against the stored chunk vectors using a similarity measure (e.g., cosine similarity). The retriever returns the top-k most similar chunks together with their similarity scores and provenance metadata (document identifier and page reference). These retrieved units constitute the evidence set that will be formatted and injected into the prompt for generation.

Once the top-k chunks are retrieved, their textual content is concatenated into a single context string that is injected into the prompt used for generation. The context construction step is not a mere formatting detail: it determines which evidence the LLM can access and how easily the model can reference it. In this

system, each chunk is formatted with an explicit reference identifier and with provenance metadata that includes the source paper and page number. This design supports traceability by enabling the answer to reference evidence through stable chunk identifiers while preserving the link to the original document. The end-to-end retrieval stage is executed as the first step of the interactive RAG query pipeline. In the code, the retrieved chunks are used to build the context and the final prompt before invoking the LLM backend. This separation between retrieval and generation makes it possible to study retrieval parameters such as k independently from generation settings, and it supports systematic evaluation across configurations.

4.6 Generation layer and prompt templates

The generation layer transforms the retrieved evidence set into a natural-language answer. In the proposed system, generation is controlled through prompt templates that specify how the LLM should consume the retrieved context, how it should cite evidence, and how it should behave when the context does not support an answer. Prompt templates are implemented as first-class, configurable artifacts in llm file, enabling systematic experimentation across different prompting strategies.

4.6.1 Prompt template catalogue

The system defines five prompt templates, each representing a different trade-off between verbosity, strictness, and reasoning style. Table 4.3 summarises the intent of each template. Template 1 is the most structured and is used as default, enforcing inline citations and a final `References` section. Templates 2 and 5 provide lighter guidance, while Template 4 imposes strict abstention rules when the answer cannot be supported by the retrieved chunks. Template 3 explicitly encourages step-by-step reasoning prior to producing a cited answer.

Table 4.3: Prompt templates implemented in `llm.py`.

ID	Name	Purpose and constraints
1	Detailed	Structured academic assistant; uses only provided chunks; enforces inline citations `[i]` and a final `References` section matching cited IDs.
2	Minimal	Short instruction to answer using only the context and cite sources; allows a brief "cannot find" response.
3	Step-by-step	Encourages relevance analysis and step-by-step reasoning before producing a final answer with citations.
4	Strict rules	Enforces abstention if the answer is not supported; every claim must have a citation; forbids external knowledge.
5	Conversational	More natural tone while still requiring citation markers when using chunk information; encourages honesty under missing evidence.

Templates are stored in a dictionary keyed by template ID. Listing 4.1 reports the full prompt template number 1. The whole templates are available in the repository.

Listing 4.1: First Prompt template registry.

```
You are an academic Research Assistant.
Your goal is to answer the user's question using ONLY the provided
context chunks.

CONTEXT STRUCTURE:
The context consists of chunks, each labeled with a specific ID
(e.g., "--- REF ID: [1] ---"). Each chunk also contains metadata:
(Source: filename, Page: number).

INSTRUCTIONS:
1. Answer: Synthesize the information to answer the question.
2. Grounding: Use ONLY information from the chunks.
3. Inline Citations: You MUST cite the ID immediately after using
   information, using square brackets: [1], [2].
4. References Section: At the very end, create "References".
   List ONLY the IDs you actually cited.

OUTPUT FORMAT:
<Your answer text with inline citations like [1], [3]...>

```

```
References:
[1] Source: <filename>, Page: <page>
[3] Source: <filename>, Page: <page>
```

4.6.2 Closed vs. open-knowledge prompting

In addition to template selection, the system supports an `open_knowledge` flag that modifies the system prompt by appending an extra instruction block. When `open_knowledge=True`, the model is allowed to complement document-grounded information with general knowledge, but it is instructed to distinguish between chunk-based content (which must be cited) and external knowledge (which should not be cited). Listing 4.2 shows the suffix added to the selected template.

Listing 4.2: Open-knowledge instruction suffix.

```
OPEN_KNOWLEDGE_SUFFIX = """
ADDITIONAL INSTRUCTION:
You may use your general knowledge to complement the information
from the chunks if it helps provide a more complete answer.
However, clearly distinguish between information from the chunks
(cite with [1], [2]) and your own knowledge (no citation needed).
"""
```

Template selection and suffix composition are implemented in `get_prompt_template`.

4.6.3 Prompt construction and LLM invocation

The final prompt combines three elements: the selected system prompt (template), the retrieved context string, and the user question. Once the prompt is built, the system queries the selected LLM backend. In the local configuration, generation is performed through an Ollama server by calling the `/api/generate` endpoint, with a configurable model name and a context window parameter (`num_ctx`) set to 4096 in the default configuration. There is also Azure OpenAi, requests are sent to the Azure Chat Completions endpoint, authenticated via an API key passed in the api-key header. The deployment name (e.g., gpt-4o) is embedded in the URL path. The prompt is wrapped in a single user message, as the system prompt is already concatenated into the prompt text. This setup enables running the system fully

locally while keeping the generation interface compatible with the experimental pipeline described in Chapter 5.

4.7 Operating modes and configuration management

The implemented system is designed to support two complementary operating modes. The first mode targets interactive exploration of the indexed paper corpus, where a user can submit one or more questions and receive an answer grounded on retrieved chunks. The second mode targets systematic experimentation, where the same pipeline is executed over a benchmark dataset under multiple configurations and results are exported in a structured format for subsequent analysis. Both modes share the same core components (retrieval, prompt construction, and LLM invocation), but expose different interfaces and configuration knobs.

4.7.1 Interactive query mode

The interactive mode provides a lightweight interface for querying the corpus in real time. It performs one-shot retrieval, constructs a prompt that includes the retrieved context, invokes the LLM, and finally prints the retrieved sources for transparency.

- initialise runtime configuration (model, top-k, operating mode);
- connect to the vector database and load the prompt template;
- for each user query, run the online QA workflow.

4.7.2 Batch evaluation mode

To enable controlled comparisons across alternative configurations, the system includes a batch evaluation mode implemented in evaluation file. This mode executes the pipeline over an internal benchmark dataset and computes metrics such as

source accuracy, page accuracy, and answer similarity. It exposes key experimental variables as command line arguments, including the LLM backend selection, retrieval depth (`top-k`), prompt template ID, and the `open_knowledge` flag. The evaluation script exposes the main experimental parameters via CLI flags, enabling reproducible runs and systematic sweeps over configurations.

If no template ID is provided, the evaluation runner executes all five prompt templates sequentially, enabling direct comparison under the same dataset and retrieval settings. Results are exported to CSV (and optionally to a detailed CSV including full answers), supporting aggregation and inspection in the analysis stage presented in Chapter 5.

4.7.3 Configuration parameters

Core parameters are centralised in config file. The configuration includes dataset paths, embedding model settings, vector database connection parameters, chunking and retrieval hyperparameters, and LLM backend settings. Several values can be overridden via environment variables, which is particularly useful in containerised deployments where endpoints and model names may vary across machines.

Table 4.4 summarises the complete design space of the experiments. These parameters are treated as first-class experimental variables in the evaluation chapter, enabling reproducible comparisons across prompt templates, retrieval depth, and operating modes.

Table 4.4: Main configuration knobs

Dimension	Fixed/Variable	Value(s)
Embedding model	Fixed	`all-MiniLM-L6-v2` (384d)
Vector DB	Fixed	Qdrant, cosine distance
Chunk size	Fixed	1000 chars
Chunk overlap	Fixed	200 chars
Top-k	Fixed	5
Temperature	Fixed	0.6
Context window	Fixed	4096 tokens
LLM model	Variable	`llama3.2`, `qwen2.5:7b`, `gpt-4o`
Prompt template	Variable	1, 2, 3, 4, 5
Open knowledge	Variable	No, Yes
Seed	Variable	0, 1, 2

Overall, providing both an interactive mode and an evaluation mode, together with centralised configuration and container-friendly environment overrides, supports the dual goal of the project: delivering a usable prototype for paper-grounded question answering and enabling systematic experimentation on how engineering choices affect answer quality and traceability.

4.8 Evaluation pipeline and experiment logging

In addition to the interactive querying workflow, the system includes a dedicated evaluation pipeline designed to run reproducible experiments over an internal benchmark dataset and to export results in a structured format. The evaluation code is implemented in evaluation file and reuses the same core building blocks of the interactive pipeline (retrieval, context formatting, prompt construction, and LLM invocation), ensuring that batch experiments reflect the behaviour of the deployed system.

4.8.1 Benchmark dataset interface

The evaluation runner consumes a benchmark dataset stored as a JSON file. The dataset loader reads the file into memory and provides a list of question records. The benchmark dataset is loaded from a structured file and converted into an in-memory list of test queries. Each entry includes the query text and its associated metadata (e.g., category label and expected answerability), which are later used to stratify results during evaluation. This separation between dataset definition and execution logic enables reproducible runs and simplifies the extension of the benchmark with additional queries.

4.8.2 Evaluation execution and configuration grid

The entry point of the evaluation script exposes the main experimental variables through a command line interface. In particular, the user can control the random seed, the LLM backend model name, the retrieval depth (`top-k`), the prompt template ID, and the `open_knowledge` flag. If no specific template is provided, the evaluation runner executes the full set of five templates sequentially, enabling direct comparison under the same dataset and retrieval settings.

A key feature of the implementation is the ability to evaluate not only the retrieved evidence but also the evidence effectively used by the model. When the `--cited-only` flag is enabled, the evaluation considers only the chunks that are explicitly cited by the model in the generated response. This behaviour is supported by parsing citation markers of the form `[1]`, `[2]`, etc. in the LLM output and mapping them back to the retrieved chunk list.

4.8.3 CSV export and logging format

To support analysis and aggregation of experimental results, the evaluation pipeline exports outcomes to CSV files. The system produces a standard CSV containing compact metrics for each question and configuration, and an optional detailed CSV that also includes the full question text, retrieved sources/pages, and the complete LLM response. Both exports use a semicolon delimiter (`;`) to avoid ambiguities in fields that may include commas.

How it works the CSV export function? Each row records the configuration parameters (`seed`, `llm`, `template_id`, `open_knowledge`), the question identifier, and the computed metrics (`source_accuracy`, `page_accuracy`, `similarity`). The export logic supports either appending to an existing file (default) or overwriting when the `--overwrite` flag is provided. When appending, an empty line is inserted between blocks of results to improve readability.

Table 4.5 summarises the schema of the exported standard CSV, which is the primary artefact used for quantitative analysis in Chapter 5. The detailed export follows the same principle but includes additional diagnostic fields, enabling fine-grained error analysis (e.g., inspecting which sources and pages were retrieved and which answer text was produced under a given configuration).

Table 4.5: Schema of the standard evaluation CSV export.

Column	Type	Description
`seed`	int	Seed value recorded for the run
`llm`	string	LLM backend model identifier (e.g., Ollama model name)
`template_id`	int	Prompt template identifier (1–5)
`open_knowledge`	string	Flag recorded as `Yes`/`No`
`question_id`	string/int	Identifier of the benchmark question
`source_accuracy`	float	Source-level correctness score
`page_accuracy`	float	Page-level correctness score
`similarity`	float	Cosine similarity between expected and generated answers

Overall, this evaluation pipeline operationalises the experimental requirements defined in Chapter 3. It enables systematic comparisons across prompt templates, retrieval depth, and operating modes, and it produces structured logs that support both aggregate analysis (via the standard CSV) and detailed inspection of failure cases (via the detailed CSV export). The next section describes the container-based deployment that makes these experiments reproducible across environments.

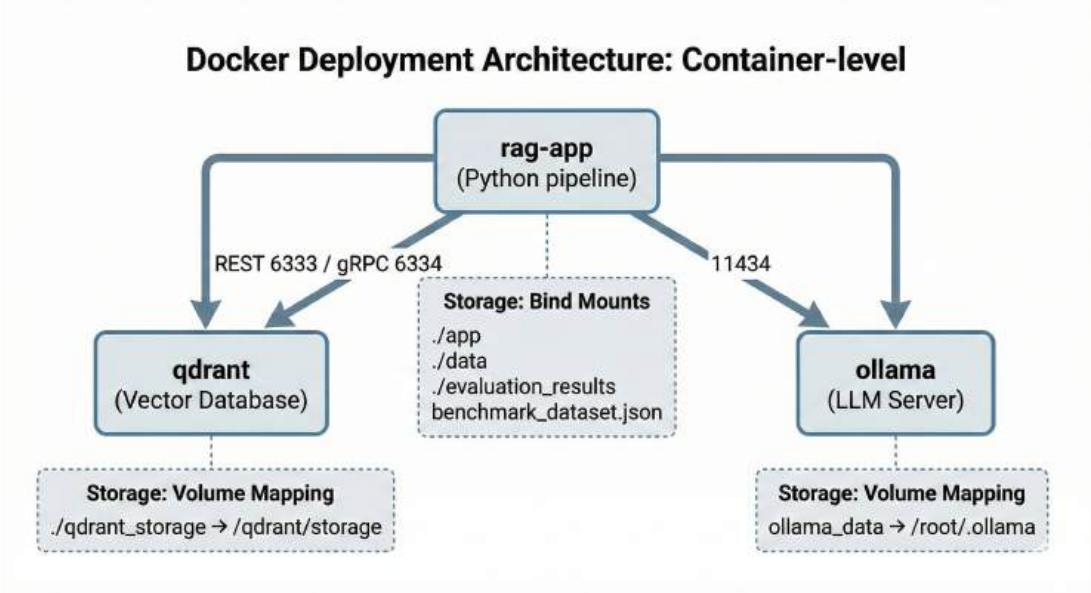

Figure 4.2: Container-level deployment architecture of the system.

4.9 Deployment and reproducibility with Docker

To ensure portability and reproducibility, the system is deployed using Docker. Containerisation provides a consistent runtime environment for all pipeline components, reduces machine-specific dependency issues, and simplifies replication of experiments. This is particularly relevant for this project, where the same pipeline must support interactive querying and repeated batch evaluations under multiple configurations.

Figure 4.2 shows the container-level deployment architecture. The `rag-app` container orchestrates the pipeline and communicates with Qdrant for similarity search and with Ollama for local LLM inference. Persistent storage is managed through a bind mount for Qdrant data and a named volume for Ollama models, while the application container uses bind mounts for the codebase, the PDF corpus, the benchmark dataset, and the evaluation outputs.

Table 4.6 summarises the services defined in the Docker Compose setup and their responsibilities. Qdrant stores the embedding index and supports top-k similarity search. Ollama hosts the local LLM models and exposes a generation API. The `rag-app` service builds the project image and runs the pipeline, including

ingestion, retrieval, interactive querying, and batch evaluation.

Table 4.6: Docker Compose services and responsibilities in the deployment.

Service	Component	Responsibility
`qdrant`	Vector database	Stores chunk embeddings and provenance payload; provides REST (6333) and gRPC (6334) APIs for similarity search
`ollama`	LLM server	Hosts local LLM models and exposes an HTTP API (11434) for generation
`rag-app`	Python pipeline	Runs ingestion, indexing, retrieval, interactive queries, and batch evaluation; exports CSV logs

The setup explicitly models the system architecture through service separation and dependency ordering. Configuration is provided through environment variables that align with config file, including hostnames for service discovery inside the Docker network, the corpus path, the default local model name, and optional Azure OpenAI parameters. The application container mounts the project directories in real time, enabling rapid iteration during development, and it mounts an output directory for storing evaluation results.

From a reproducibility standpoint, Docker contributes in three ways. First, it standardises the versions of external services used by the pipeline (Qdrant and Ollama), ensuring that retrieval and generation interfaces remain stable across machines. Second, it centralises configuration through environment variables, making experiments easier to document and rerun under the same settings. Third, it separates persistent artefacts from ephemeral containers: vector index data and LLM model weights are stored in dedicated mounts/volumes, while evaluation outputs are exported to a host directory. Together, these elements support repeatable evaluation runs and facilitate systematic experimentation across the configuration grid.

Chapter 5

Results and evaluation

5.1 Experimental setup

This chapter reports the experimental evaluation of the proposed RAG system. The evaluation is designed to quantify how key engineering choices influence the system behaviour in an academic paper question answering scenario. In particular, experiments compare alternative LLM backends and prompt templates, and analyse the effect of enabling an open-knowledge prompting mode versus a corpus-only mode. All experiments are executed on the same indexed corpus and use fixed preprocessing, chunking, and retrieval settings to ensure comparability. To get more in deep this is the project repository: `https://github.com/AlessandroDeFaveri/RAG-System-Project.git`.

5.1.1 Corpus and indexing snapshot

The evaluation uses the local corpus described in Chapter 2, consisting of 50 faculty-authored full papers in English. Each paper is ingested through the preprocessing and chunking pipeline described in Chapter 4. Chunking is performed page-by-page with a punctuation-aware splitting heuristic, using `CHUNK_SIZE = 1000` and `CHUNK_OVERLAP = 200`. Each chunk is embedded with `all-MiniLM-L6-v2` and indexed in Qdrant together with provenance metadata (`source`, `page`, and a sequential chunk identifier). This indexing snapshot remains fixed across all evaluation runs.

5.1.2 Benchmark dataset

Experiments are conducted on an internal benchmark dataset containing 10 questions(5 in-scope, 3 out-of-scope, 2 multi-paper). Each benchmark record includes a natural-language query and reference information used for evaluation, including an expected answer and ground-truth provenance fields (expected source document and expected page). The benchmark is designed to be answerable from the indexed paper corpus and to require document grounding rather than generic domain knowledge.

5.1.3 Compared configurations

The evaluation compares configurations obtained by combining three LLM backends (`llama3.2`, `qwen2.5:7b`, and `gpt-4o`) with five prompt templates (IDs 1–5) and two operating modes. In the corpus-only setting (`open_knowledge = No`), the prompt instructs the model to rely exclusively on retrieved chunks. In the open-knowledge setting (`open_knowledge = Yes`), the prompt allows the model to complement chunk-based evidence with general knowledge while preserving citation requirements for chunk-derived content. Retrieval depth is fixed to `TOP_K = 5` across all configurations.

To support repeatable comparisons, experiments are executed with controlled seeds where applicable and with structured logging. Each run produces a detailed CSV output that records the configuration parameters (LLM, template ID, open-knowledge flag, and seed), the question identifier, the retrieved sources and pages, the generated answer, and the evaluation scores computed by the pipeline. This detailed logging enables both aggregate analysis (averaging results across questions and runs) and qualitative inspection of failure cases.

5.2 Evaluation metrics

The evaluation pipeline computes three metrics for each question and configuration. The metrics are chosen to reflect the core requirements of the case study: correctness of provenance, strict traceability at page level when available, and alignment of the generated answer with an expected reference.

At first, source-level accuracy measures whether the system identifies the correct supporting paper for a given query. Each benchmark record specifies an expected source document, and the pipeline compares it against the provenance returned by the system. Depending on the evaluation setting, provenance can be derived from the retrieved set or from the subset of chunks that the model explicitly cites in the final answer. The metric is reported in the range $[0, 1]$, where 1 indicates that the expected paper is correctly matched and 0 indicates a mismatch.

Moreover, page-level accuracy measures whether the system identifies the correct page of the supporting evidence when page provenance is available. Each benchmark record specifies an expected page, and the pipeline compares it against the page values associated with the system provenance. This metric directly operationalises the strict traceability preference discussed in Chapter 2: when the system is able to provide page information, page references should be correct and should enable efficient verification within the original PDF. The metric is reported in $[0, 1]$ and is interpreted jointly with source-level accuracy, since page attribution is meaningful only when the correct source document is identified.

Therefore, answer similarity measures semantic alignment between the generated answer and the expected reference answer provided in the benchmark dataset. The pipeline computes cosine similarity in an embedding space, producing a score in $[0, 1]$, where larger values indicate closer semantic correspondence. Similarity captures whether the answer content matches the reference at a conceptual level, but it does not, by itself, guarantee groundedness at the claim level. For this reason, similarity is interpreted together with provenance-based metrics, which provide evidence that the answer is linked to the correct source and (when available) to the correct page.

The three metrics support complementary perspectives on system behaviour. Source accuracy and page accuracy quantify the provenance correctness required for verification, while similarity quantifies answer-content alignment with the benchmark reference. In open-knowledge mode, similarity should be interpreted with additional caution: allowing general knowledge can improve fluency and completeness, but it can also introduce content that is not present in the reference answer and therefore reduce similarity even when the response remains useful. For this reason, the analysis in the next section reports results by configuration and

discusses differences across operating modes and prompt templates.

5.3 Quantitative results

This section reports aggregated results across the evaluated configuration grid. Unless otherwise stated, reported values are averages computed over all benchmark questions and all runs available for a given configuration group.

5.3.1 Overall performance by LLM backend and operating mode

Table 5.1 reports average performance grouped by LLM backend and operating mode. Source accuracy is stable across all groups (this is expected because the metric depends exclusively on the retrieval stage, which is model-independent given the same embedding and top-k setting.), indicating that, on this benchmark, the system identifies the correct supporting paper in most cases regardless of the backend and mode. Differences emerge primarily in answer similarity and, to a lesser extent, in page-level accuracy.

Table 5.1: Overall results by LLM backend and operating mode (averaged across templates and seeds).

LLM	open_knowledge	n	Source acc.	Page acc.	Similarity
gpt-4o	No	100	0.800	0.437	0.489
llama3.2	No	100	0.800	0.403	0.516
qwen2.5:7b	No	100	0.800	0.403	0.550
llama3.2	Yes	100	0.800	0.403	0.496
gpt-4o	Yes	100	0.800	0.453	0.536
qwen2.5:7b	Yes	100	0.800	0.403	0.537

Enabling open knowledge increases similarity for `gpt-4o` while leaving source accuracy unchanged and slightly improving page accuracy. For the local models (`llama3.2` and `qwen2.5:7b`), open knowledge decreases similarity on average, suggesting that allowing general knowledge may introduce additional content that

diverges from the benchmark reference answers. Regarding similarity, the only LLM-dependent metric, tells a more interesting story: the mid-size qwen2.5:7b outperforms both the smaller llama3.2 and the much larger gpt-4o. This suggests that, within a constrained RAG pipeline (fixed context, strict prompts, 4 096-token window), a well-tuned 7 B model can exceed a general-purpose proprietary model whose strengths(multilingual fluency, broad world knowledge) are largely neutralised by the grounded-generation constraint.

5.3.2 Impact of prompt templates

To isolate the effect of prompting, Table 5.2 reports the average answer similarity obtained by each prompt template for each LLM backend. The table also reports the overall average across models. The results show consistent differences across templates, indicating that prompting acts as a controllable lever for answer formulation and alignment with reference answers.

Table 5.2: Average answer similarity by prompt template and LLM backend (higher is better).

Template	llama3.2	qwen2.5:7b	gpt-4o	Avg
T1 Detailed	0.527	0.529	0.485	0.514
T2 Minimal	0.498	0.512	0.428	0.479
T3 Chain-of-Thought	0.467	0.498	0.514	0.493
T4 Strict	0.547	0.671	0.621	0.613
T5 Conversational	0.492	0.508	0.513	0.504

Across all backends, T4 (Strict) achieves the highest average similarity (0.613), with a particularly strong effect on `qwen2.5:7b` (0.671) and `gpt-4o` (0.621). This suggests that stricter prompting constraints encourage responses that remain closer to the evidence and reduce unconstrained elaborations, which improves alignment with the benchmark references. Conversely, T2 (Minimal) yields the lowest average similarity (0.479), indicating that lighter instructions may be insufficient to consistently steer the model toward the expected level of specificity and grounding. T3 (Chain-of-Thought) shows heterogeneous behaviour: it improves similarity for

`gpt-4o` compared to T1, but underperforms for the local backends, suggesting that step-by-step prompting may not provide uniform gains across models in this setting.

5.3.3 Performance by question category

Beyond configuration-level averages, it is useful to examine performance by question category, as different query types impose different demands on retrieval and generation. Table 5.3 reports the average answer similarity by category for each LLM backend. The benchmark questions are grouped into three categories: in-scope single-paper questions (Q1–Q5), out-of-scope questions (Q6–Q8), and multi-paper questions (Q9–Q10). The reported N values reflect the number of evaluated instances per category across the configuration grid.

Table 5.3: Average answer similarity by question category and LLM backend.

Category	llama3.2	qwen2.5:7b	gpt-4o
In-scope (Q1–Q5, $N = 100$)	0.698	0.738	0.674
Out-of-scope (Q6–Q8, $N = 60$)	0.131	0.202	0.245
Multi-paper (Q9–Q10, $N = 40$)	0.590	0.569	0.509

As expected, similarity is highest for in-scope questions, where the corpus contains the necessary evidence and retrieval can supply relevant chunks. In this category, `qwen2.5:7b` achieves the strongest average similarity (0.738), followed by `llama3.2` (0.698) and `gpt-4o` (0.674). Out-of-scope questions yield very low similarity (0.13–0.24). These questions ask about topics not covered in the indexed papers; the expected answer is a variant of "I don't know." The low scores arise from a lexical mismatch: models produce varied refusal phrasings that diverge from the short reference string. Interestingly, gpt-4o scores highest here (0.245), suggesting its refusal phrasing is more consistently close to the reference than the smaller models' attempts.

Multi-paper questions show intermediate performance and are generally more challenging than single-paper in-scope questions. These queries require integrating information across multiple documents and therefore increase the likelihood of retrieval incompleteness or partial evidence. The results suggest that `llama3.2` and

`qwen2.5:7b` perform comparably in this setting (0.590 and 0.569), while `gpt-4o` attains lower similarity (0.509), indicating that cross-document synthesis remains a challenging regime in the evaluated setup.

Overall, the quantitative results indicate that prompting choices significantly affect answer similarity (with the strict template yielding the highest alignment), and that performance varies substantially across question categories. In particular, out-of-scope and multi-paper questions represent the most challenging regimes in the benchmark. The next section therefore complements aggregate metrics with qualitative error analysis, focusing on failure cases related to insufficient evidence handling, citation behaviour, and cross-document synthesis.

5.3.4 Error Analysis

Quantitative metrics summarise overall trends, but they do not fully explain why failures occur. This section analyses recurring error patterns observed in the detailed evaluation outputs, with emphasis on the most challenging regimes identified in Section 5.3.3: out-of-scope questions and multi-paper questions. The goal is to characterise the dominant failure modes at the level of retrieval, prompting, and citation behaviour, and to connect them to the design choices discussed in Chapter 4.

1. *Insufficient-evidence handling.* Out-of-scope questions are expected to trigger a brief refusal such as *"I don't know based on the provided documents."* All template configurations do produce refusals on these questions; however, the form of the refusal varies significantly. Under strict prompting (Template 4), the system consistently generates short, formulaic statements that closely match the expected answer (cosine similarity ≈ 0.88). Under more permissive templates, the system still refuses but produces longer responses that explain why the context is insufficient or describe what the retrieved passages actually contain, resulting in substantially lower similarity scores ($\approx$ 0.05–0.13) despite the behaviour being qualitatively correct. When open knowledge is enabled, some models occasionally go further and attempt to answer from general knowledge—for instance, `llama3.2` provides a Carbonara recipe on Q6, and `gpt-4o` generates cooking instructions on

the same question. These extrapolations *decrease* semantic similarity because the expected answer is a refusal, not a substantive response. From a traceability perspective, this highlights a trade-off: permissive prompting can produce helpful content, but it undermines the requirement that claims be supported by the indexed papers.

2. *Page attribution.* Even when the correct paper is retrieved and cited, page references are frequently incorrect. This pattern is most visible for Q2 and Q4, where source accuracy is 1.0 but page accuracy is 0.0 across all models and seeds. The primary cause is that the chunking algorithm may assign a passage to a page range that does not match the page expected by the benchmark, either because the relevant content spans a chunk boundary or because the chunk was extracted from a neighbouring page. Additionally, when the evidence spans multiple chunks, the model may cite the chunk that contains supporting text rather than the one corresponding to the benchmark's expected page. These issues explain why page-level accuracy (0.42 overall) is systematically lower than source-level accuracy (0.80): page correctness is a stricter requirement that depends on the stability of PDF extraction, segmentation boundaries, and the alignment between chunk-level metadata and the benchmark's ground truth.

3. *Retrieval noise and partial relevance.* In several cases, the top-k evidence set contains passages that are topically related but not directly supportive of the specific question. For Q3, the retriever consistently returns the wrong source document (source accuracy = 0.0), yet models still attempt to answer, sometimes producing plausible-sounding but incorrectly grounded responses. When the prompt is permissive, the model may synthesise an answer from weakly related context without signalling uncertainty. Strict prompting mitigates this by forcing the model to stay closer to the provided context and to cite specific references, which aligns with the higher similarity observed for Template 4. However, strict prompting can also produce false negatives when the retrieved evidence is only partial: on multi-paper questions Q9 and Q10, `gpt-4o` with Template 4 refuses to answer entirely (similarity ≈ 0.02–0.05) even though the retriever has found one of the two expected sources

(source accuracy = 0.5). This reveals a sensitivity to retrieval completeness that is template-dependent.

4. *Multi-paper questions.* These queries require integrating evidence across multiple documents. The retriever typically surfaces one of the two expected sources (source accuracy = 0.5 for both Q9 and Q10), but not both. Three distinct failure modes are observed: (a) the model provides a partial answer grounded in a single paper, covering only part of the expected comparison—this is the most common behaviour for `llama3.2` and `qwen2.5:7b`; (b) the model attempts synthesis across papers but produces incomplete or loosely supported comparisons, citing references from the single retrieved source; and (c) the model refuses to answer entirely, as seen with `gpt-4o` under strict prompting, where the partial evidence is deemed insufficient. These failures are consistent with a known limitation of single-stage retrieval systems: when the answer requires evidence aggregation from multiple documents, a fixed top-k pipeline may not retrieve the necessary breadth of sources.

5. *API failures and language inconsistency.* A non-negligible fraction of `gpt-4o` responses (11 out of 200, 5.5%) are connection errors returned by the Azure OpenAI endpoint, concentrated in seed 1 with Templates 1 and 2. These failures produce very low similarity scores and contribute to the lower overall average for `gpt-4o`. Additionally, 7 out of 200 `gpt-4o` responses (3.5%) are generated in Spanish rather than English, despite the prompts and expected answers being in English. This language inconsistency further penalises the cosine similarity metric, as the embedding model computes lower similarity between semantically equivalent sentences in different languages. Both issues are external to the RAG pipeline design but affect the reliability of cloud-based model integration.

Finally, the analysis reveals that *prompt templates shape how failures manifest* rather than whether they occur. On out-of-scope questions, all templates produce refusals, but strict templates yield concise responses that align closely with the expected answer, while permissive templates produce verbose explanations that reduce measured similarity despite being qualitatively correct. On in-scope questions, strict templates encourage tighter citation discipline and reduce unsupported

elaboration, improving alignment with reference answers. On multi-paper questions, strict prompting can be counterproductive by triggering false refusals when evidence is only partially available. This observation supports treating prompting as a first-class experimental variable in RAG pipelines, particularly in academic settings where traceability and verifiability are core requirements, and underscores the need to interpret similarity-based metrics in light of the response style imposed by each template.

Overall, the error analysis highlights three core insights: (i) prompting primarily controls the *shape* of failure (refusal style vs. speculative answers) rather than eliminating it; (ii) provenance precision, especially at the page level, is the most fragile link in the pipeline and is highly sensitive to extraction and chunking boundaries; and (iii) single-stage retrieval is insufficient for questions that require evidence aggregation across documents, making retrieval breadth a dominant factor in multi-paper performance. These insights motivate future improvements that target provenance robustness and retrieval coverage, alongside prompt designs that balance strict grounding with graceful handling of partial evidence.

5.4 Discussion and threats to validity

The experimental results provide a controlled comparison of key design variables of the proposed RAG system, but their interpretation is subject to several limitations that should be made explicit.

A first threat to validity concerns the size and composition of the benchmark dataset. The evaluation is conducted on 10 questions, which enables detailed inspection of configurations and failure cases but limits statistical power. A small benchmark may not cover the full diversity of query types encountered in realistic usage, and averages may be sensitive to a small number of difficult questions. Future evaluations should expand the dataset and include a more systematic balance between definition-seeking, method-focused, results-focused, and comparison-oriented questions.

A second threat concerns the dependence on PDF extraction and on page-level alignment. Page accuracy is a strict metric that requires stable provenance mapping from extracted text to page indices. Extraction artefacts, layout vari-

ability across papers, and segmentation boundaries can all affect page attribution independently of whether the correct evidence is present in the corpus. As a result, page-level errors may reflect preprocessing limitations rather than purely retrieval or generation failures. This limitation is intrinsic to PDF-based corpora and motivates considering complementary provenance signals (e.g., section headers or paragraph identifiers) when page mapping is unreliable.

A third threat concerns the evaluation metric used for answer similarity. Similarity is computed in an embedding space and captures semantic closeness between generated and reference answers, but it does not directly measure groundedness at the claim level. Two answers can be semantically similar while differing in subtle but important factual details, and a grounded answer can receive a lower similarity score if it uses a different phrasing or level of detail than the reference. For this reason, similarity should be interpreted together with provenance metrics (source and page accuracy) and supported by qualitative inspection, as done in the error analysis section.

A fourth threat concerns the interpretation of open-knowledge mode. Allowing general knowledge can improve completeness and readability, but it can also introduce content that is not present in the benchmark reference answers, reducing similarity even when the response remains useful. More importantly, open knowledge complicates the notion of groundedness: a response may be correct in a general sense while not being strictly attributable to the local corpus. Consequently, comparisons between closed and open modes should be interpreted with respect to the intended usage scenario. In academic workflows that prioritise verifiability and citation-based navigation, closed-mode behaviour and conservative prompting may be preferable even when open mode yields higher similarity for some configurations.

Despite these limitations, the evaluation supports two robust conclusions within the tested scope. First, prompt design has a measurable impact on answer similarity and on conservative behaviour under insufficient evidence, with strict prompting achieving the highest alignment in the benchmark. Second, performance varies strongly across question categories, confirming that out-of-scope and multi-paper queries represent the most challenging regimes. These observations motivate the future-work directions discussed in the conclusion chapter, includ-

ing improvements on dataset design, provenance robustness, and prompt-driven control mechanisms for traceability.

Chapter 6

Conclusion and future work

6.1 Summary of the problem and approach

This thesis addressed the problem of enabling reliable question answering over a local corpus of academic papers authored by university faculty. In this setting, answer usefulness depends not only on fluency but on verifiability: users must be able to trace key statements back to the underlying documents and, when possible, to the page where supporting evidence appears. This requirement motivates the adoption of Retrieval-Augmented Generation (RAG), where retrieval over an indexed document collection provides explicit evidence that conditions the output of a generative language model.

To meet these requirements, the thesis presented an end-to-end RAG system that ingests scientific PDFs, extracts page-level text, segments it into retrievable chunks, computes dense embeddings with `all-MiniLM-L6-v2`, and indexes the resulting vectors in Qdrant together with provenance metadata. At query time, the system retrieves the top-k most similar chunks (with `TOP_K=5`), constructs a prompt that injects the retrieved evidence as context, and generates an answer using either a local LLM backend (Ollama with `llama3.2` and `qwen2.5:7b`) or an optional cloud backend (Azure OpenAI). The system supports both interactive usage and systematic evaluation through a batch pipeline that exports results to CSV for comparative analysis.

A central aspect of the approach is prompt engineering. Instead of treating

prompting as a fixed interface detail, the system implements multiple prompt templates that represent different levels of strictness and reasoning style, including a strict template designed to enforce evidence-based answering and conservative behaviour under insufficient evidence. The evaluation framework compares these templates across backends and operating modes (corpus-only vs. open knowledge) to quantify how prompt design influences answer quality, provenance correctness, and robustness to challenging query categories.

6.2 Main findings

The experimental evaluation supports several findings about the behaviour of the proposed system under the tested configurations. First, prompt design has a measurable impact on answer formulation and on alignment with the benchmark references. Across backends, the strict prompt template (T4) achieved the highest average similarity (0.613), outperforming more permissive templates such as the minimal template (T2, 0.479). This result indicates that, in this setting, stricter instructions reduce unconstrained elaboration and encourage answers that remain closer to the retrieved evidence and to the expected level of specificity.

Second, the effect of allowing open knowledge differs across LLM backends. For `gpt-4o`, enabling open knowledge increased average similarity while preserving provenance metrics, suggesting that the model can complement corpus-grounded information without substantially diverging from the benchmark references. For the local backends (`llama3.2` and `qwen2.5:7b`), open knowledge reduced similarity on average, consistent with a tendency to add information not present in the reference answers. This behaviour reinforces the interpretation that open knowledge can be beneficial when it is used to clarify or contextualise retrieved evidence, but it may be counterproductive when it introduces content that is difficult to reconcile with strict document-grounded evaluation.

Third, performance varies strongly by question category. In-scope questions (Q1–Q5) achieved the highest similarity across backends, with `qwen2.5:7b` reaching 0.738 on average, followed by `llama3.2` (0.698) and `gpt-4o` (0.674). Out-of-scope questions (Q6–Q8) yielded substantially lower similarity for all models, confirming that insufficient-evidence scenarios remain challenging and that bench-

mark alignment is difficult when the requested information is not present in the corpus. Multi-paper questions (Q9–Q10) produced intermediate similarity scores, reflecting the additional difficulty of aggregating evidence across multiple documents within a fixed retrieval depth and context budget.

Finally, provenance correctness exhibits different levels of difficulty. Across the evaluated configurations, source-level correctness is comparatively stable, while page-level correctness is lower. This gap is consistent with the stricter nature of page attribution, which depends on stable PDF extraction and on segmentation boundaries, and it highlights that page-level traceability is a harder target than document-level traceability in PDF-based pipelines. Together, these findings confirm that the proposed architecture is effective at retrieving and attributing the correct documents in most cases, while answer formulation and strict traceability are strongly influenced by prompting choices and by the category of the query.

6.3 Limitations

The conclusions of this thesis should be interpreted in light of several limitations of the experimental setting and of the implemented pipeline.

A first limitation concerns the benchmark size. The evaluation dataset contains 10 questions, which supports controlled comparisons and detailed inspection but limits statistical power and coverage. The observed averages may be sensitive to a small number of particularly easy or difficult questions, and the benchmark may not fully represent the diversity of information needs that arise in realistic academic usage, such as broader exploratory queries, partially specified questions, or queries requiring deeper cross-document reasoning.

A second limitation concerns reliance on PDF extraction and page-level alignment. The system is designed to preserve page provenance by extracting text page-by-page and propagating page identifiers through chunking and indexing. Nevertheless, page-level accuracy remains sensitive to extraction artefacts, layout variability, and segmentation boundaries. In practice, evidence may span multiple pages or be split across chunk boundaries, and the retrieved chunk that best matches the query may not correspond exactly to the page expected by the benchmark. As a result, page-level errors may reflect the practical constraints of PDF

processing rather than purely retrieval or generation failures.

A third limitation concerns the interpretation of the similarity metric. Similarity is computed using embedding-based cosine similarity between generated and expected answers. This provides a convenient semantic measure, but it does not guarantee claim-level faithfulness or groundedness. Two answers may be semantically similar while differing in subtle factual details, and a grounded answer may score lower if it is phrased differently or uses a different level of detail than the reference. For this reason, similarity should be interpreted together with provenance-based metrics and supported by qualitative inspection, especially for out-of-scope and multi-paper questions.

A final limitation concerns open-knowledge mode. Allowing the model to use general knowledge can improve completeness or readability, but it also complicates strict attribution: even when citations are provided for chunk-based claims, the boundary between document-grounded content and external knowledge is enforced only at the prompt level and cannot be guaranteed mechanically. Consequently, results comparing open and closed modes should be interpreted with respect to the intended use: in settings where verifiability is the primary goal, conservative corpus-only behaviour may be preferable even when open knowledge yields higher similarity under some conditions.

6.4 Future work: prompt engineering for controllability and traceability

Future work will primarily focus on prompt engineering as a mechanism to improve controllability, traceability, and robustness of the proposed RAG pipeline. The empirical results of this thesis indicate that prompt choice materially affects answer similarity and failure expression, especially under insufficient evidence and multi-paper settings. Related work also supports the view that prompting is not merely a surface-level interface, but a systematic design space that can be explored, structured, and optimised.[12]

1. A first direction is *category-aware prompting*. The benchmark analysis shows clear differences across question categories (in-scope, out-of-scope, and multi-

paper), suggesting that a single static prompt may not be optimal across regimes. A practical extension is to introduce a lightweight query classifier that selects a prompt template (or modifies prompt constraints) based on the predicted category. For example, out-of-scope queries could trigger a stricter abstention-oriented template, while multi-paper queries could trigger a template that explicitly instructs the model to aggregate evidence across multiple sources and to cite multiple references when required. This approach is aligned with prompt engineering practice, where prompt structure and constraint strength are tailored to the task and risk profile.[12]

2. A second direction is *automatic prompt selection and optimisation*. In the current system, template choice is a manual configuration knob. Future work could learn a template-selection policy from evaluation logs by treating prompt selection as a decision problem, optimising for a weighted objective that includes similarity, source/page accuracy, and abstention correctness. More advanced approaches could incorporate test-time optimisation loops that refine the prompt or the answer iteratively. In particular, recent work on retrieval-aware reasoning proposes critique-and-revision mechanisms that explicitly target misalignment between reasoning and retrieved evidence, suggesting that prompt design can be combined with an additional control layer at inference time.[6]

3. A third direction is *stronger citation discipline through structured outputs*. While the current templates enforce citation markers (e.g., `[1]`, `[2]`) at the text level, future work could require the model to produce a structured response schema where each claim is paired with a list of supporting reference IDs and, when available, page numbers. This would enable stricter post-generation validation: the system could check whether every citation corresponds to a retrieved chunk and whether page metadata is available, and it could reject or revise answers that violate traceability constraints. This direction is compatible with the emphasis on accurate citations in academic writing assistants, where citation correctness is treated as a first-class objective rather than a formatting detail.[4]

4. A fourth direction is *robustness-oriented prompting under retrieval noise*. Related work shows that RAG pipelines can be sensitive to misleading retrievals, producing incorrect answers that appear well grounded because they cite retrieved but unhelpful passages.[1] Prompting alone may not fully solve this issue, but future work can design prompts that (i) encourage the model to explicitly assess evidence support, (ii) request contradiction checks among retrieved passages, and (iii) prioritise abstention when evidence is conflicting. These strategies can be integrated into the strict template and evaluated specifically on out-of-scope and misleading-evidence regimes.

5. A fifth direction is to adopt *decomposition-driven prompting* for complex and multi-document queries, inspired by recent work on search-augmented reasoning such as AceSearcher.[21] AceSearcher frames search-augmented answering as an iterative process in which the model alternates between decomposing a complex query into simpler subquestions and solving them by integrating retrieved evidence, rather than relying on a single retrieval step and a single generation pass. In the context of this thesis, a lightweight variant of this idea could be implemented purely at the prompt level, by introducing a two-phase interaction pattern: first, the model generates an explicit set of subqueries or evidence requirements (e.g., "find definition", "find experimental setting", "identify limitation"), and second, it generates the final answer only after retrieval has been executed for each subquery. This approach is particularly relevant for multi-paper questions, where evidence may be distributed across documents and a single top-k retrieval call can be incomplete. Decomposition-driven prompting would therefore aim to improve coverage and reduce partial synthesis by making evidence needs explicit before answer generation.

Overall, these future-work directions treat prompt engineering as the primary lever for improving the system without radically changing the underlying architecture. They also provide a clear roadmap for extending the current prototype into a more controllable and robust assistant for academic paper question answering, where verifiability and traceability remain the dominant requirements.

Bibliography

[1] Y. Zhang, D. Yang, L. Zeng, R. Gupta, and D. Motwani, "Worse than zero-shot? a fact-checking dataset for evaluating the robustness of rag against misleading retrievals," 2025.

[2] Y. Du, M. Tian, S. Ronanki, S. Rongali, S. Bodapati, A. Galstyan, A. Wells, R. Schwartz, E. A. Huerta, and H. Peng, "Context ,length alone hurts llm performance despite perfect retrieval," Mon Oct 06 2025 21:17:13 GMT+0000 (Coordinated Universal Time).

[3] J. Sun, X. Zhong, S. Zhou, and J. Han, "Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation," 2025.

[4] Y. Wang, X. Ma, P. Nie, H. Zeng, Z. Lyu, Y. Zhang, B. Schneider, Y. Lu, X. Yue, and W. Chen, "Scholarcopilot: Training large language models for academic writing with accurate citations," *ArXiv*, vol. abs/2504.00824, 2025.

[5] A. Pachaar, "A crash course on building rag systems – part 2 (with implementations)," Nov. 2024. Daily Dose of Data Science (accessed 2026-02-15).

[6] J. Wei, H. Zhou, X. Zhang, D. Zhang, Z. Qiu, W. Wei, J. Li, W. Ouyang, and S. Sun, "Retrieval is not enough: Enhancing rag reasoning through test-time critique and optimization," Sat Oct 11 2025 02:05:28 GMT+0000 (Coordinated Universal Time).

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

[8] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih, "Dense passage retrieval for open-domain question answering," 2020.

[9] Z. Wang, C. Gao, C. Xiao, Y. Huang, S. Si, K. Luo, Y. Bai, W. Li, T. Duan, C. Lv, G. Lu, G. Chen, F. Qi, and M. Sun, "Document segmentation matters for retrieval-augmented generation," in *Findings of the Association for Computational Linguistics: ACL 2025* (W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, eds.), (Vienna, Austria), pp. 8063–8075, Association for Computational Linguistics, July 2025.

[10] Z. Zhong, H. Liu, X. Cui, X. Zhang, and Z. Qin, "Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation," *ArXiv*, vol. abs/2406.00456, 2024.

[11] C. A. Gomez-Cabello, S. Prabha, S. A. Haider, A. Genovese, B. G. Collaco, N. G. Wood, S. Bagaria, and A. J. Forte, "Comparative evaluation of advanced chunking for retrieval-augmented generation in large language models for clinical decision support," *Bioengineering*, vol. 12, no. 11, 2025.

[12] S. Schulhoff, M. Ilie, N. Balepur, K. Kahadze, A. Liu, C. Si, Y. Li, A. Gupta, H. Han, S. Schulhoff, P. S. Dulepet, S. Vidyadhara, D. Ki, S. Agrawal, C. M. Pham, G. C. Kroiz, F. Li, H. Tao, A. Srivastava, H. D. Costa, S. Gupta, M. L. Rogers, I. Goncearenco, G. Sarli, I. Galynker, D. Peskoff, M. Carpuat, J. White, S. Anadkat, A. M. Hoyle, and P. Resnik, "The prompt report: A systematic survey of prompting techniques," *ArXiv*, vol. abs/2406.06608, 2024.

[13] N. C. Ohalete, K. B. Gittner, and L. M. Matheny, "Costar-a: A prompting framework for enhancing large language model performance on point-of-view questions," Tue Oct 14 2025 15:31:21 GMT+0000 (Coordinated Universal Time).

[14] Y. Leviathan, M. Kalman, and Y. Matias, "Prompt repetition improves non-reasoning llms," 2025.

[15] B. Wen, B. Howe, and L. L. Wang, "Characterizing llm abstention behavior in science qa with context perturbations," 2024.

[16] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," 2025.

[17] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1904.09675.

[18] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: Nlg evaluation using gpt-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2511–2522, Association for Computational Linguistics, 2023. arXiv:2303.16634.

[19] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023.

[20] Z. Liu, Z. Dou, D. Lian, H. Qian, C. Gao, and Y. Wang, "Hawkbench: Investigating resilience of rag methods on stratified information-seeking tasks," 2025.

[21] R. Xu, Y. Zhuang, Z. Dong, J. Wang, Y. Yu, J. C. Ho, L. Zhang, H. Wang, W. Shi, and C. Yang, "Acesearcher: Bootstrapping reasoning and search for llms via reinforced self-play," 2025.

Acknowledgements

Optional. Max 1 page.