universität wien

Evaluation Approaches for Retrieval-Augmented Generation (RAG)

Daniel Petrov - a12028482@unet.univie.ac.at

July 23, 2025, Vienna

Contents

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a promising approach to address the limitations of Large Language Models (LLMs) by grounding outputs in external context. This work systematically evaluates a RAG application built with open-source frameworks, exploring the impact of chunking strategies, retrieval depth, and choice of generative model. To move beyond traditional metrics such as BLEU and ROUGE—which are often ill-suited for open-ended generation—and to reduce reliance on human annotation, the evaluation adopts the LLM-as-a-judge paradigm. Three frameworks—Ragas, DeepEval, and Opik—are employed, each offering diverse metrics focused on retrieval and generation quality. Results highlight both the potential and challenges of current LLM-based evaluation: while these metrics provide richer insights, they remain sensitive to model behavior and prompt design. The findings underscore the need for broader standardization to support more robust and reproducible evaluation of RAG systems.

1 Introduction

1.1 Limitations of Standalone LLMs

Large Language Models (LLMs) such as LLaMA [17], PaLM [2], and GPT [12] have shown impressive performance across a wide range of Natural Language Processing (NLP) tasks. These models encode knowledge in their parameters after exposure to large-scale training data. However, they struggle when applied to domain-specific or knowledge-intensive tasks.

A fundamental issue is the *static nature* of their internal knowledge: once trained, LLMs cannot easily incorporate new or evolving information without retraining. To address this, recent work by Ovadia et al. [13] compares two strategies for *knowledge injection*: (i) **unsupervised fine-tuning**, which updates the model's weights using *continual pretraining*, and (ii) **Retrieval-Augmented Generation (RAG)** [4, 5], which follows the principles of *in-context learning*. Their findings show that RAG consistently outperforms unsupervised fine-tuning. Unlike unsupervised fine-tuning, which is susceptible to *catastrophic forgetting* and requires significant computational resources, RAG enables the dynamic integration of external context without modifying the model's parameters.

In addition to limitations in updating their knowledge, LLMs suffer from a lack of interpretability. As these models scale to hundreds of billions of parameters, understanding how and why they generate specific outputs becomes increasingly difficult. Singh et al. [16] argue that traditional interpretability methods—such as saliency maps—do not scale effectively to such models and often fail to produce meaningful explanations. This *opaqueness* raises critical concerns in high-stakes applications such as healthcare, law, and scientific domains, where trust, safety, and transparency are paramount. Moreover, it enables a phenomenon known as *hallucination*, where models generate factually incorrect yet confidently stated information.

Recent findings by McKenna et al. [10] highlight two prominent sources of hallucinations in LLMs on downstream Natural Language Inference (NLI) tasks. The first is **attestation bias**, which refers to the model's tendency to accept a claim as true if it aligns with memorized pretraining data—even when that information is outdated, incorrect, or irrelevant to the given context. The second is **relative frequency bias**, where the model disproportionately favors common phrases observed during training, regardless of their factual accuracy. These biases can lead LLMs to ignore accurate context provided at inference time and instead default to generating incorrect or baseless answers.

To mitigate these issues, RAG offers additional benefits beyond knowledge injection: by explicitly conditioning responses on externally retrieved context, RAG introduces a form of traceability. The retrieved context not only serves as a grounding mechanism to reduce hallucinations but also improves transparency by making the source of the response evident. In this sense, RAG acts as both a corrective and interpretability-enhancing strategy.

Table 1: Summary of Standalone LLM Limitations and Mitigation via RAG

Limitation	Underlying Cause	RAG Mitigation
Static knowledge	Model parameters encode only fixed pretraining knowledge; retraining is costly	Retrieves up-to-date and domain-specific context at inference time
Catastrophic forgetting	Fine-tuning on new data can overwrite prior knowledge	RAG avoids changing model weights entirely
Lack of interpretability	Billions of parameters; hard to trace model reasoning	Retrieved context introduces traceable evidence and improves transparency
Hallucinations	Includes *attestation bias* and *relative frequency bias*	Retrieved documents help steer generation toward factually accurate responses [10]

1.2 Overview of Retrieval-Augmented Generation

To address the fundamental limitations of standalone LLMs (see Table 1), Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm. This hybrid approach combines information retrieval techniques with the generative capabilities of LLMs, resulting in more accurate, diverse, and trustworthy outputs—especially for knowledge-intensive tasks [4, 5].

At its core, a RAG system consists of two loosely coupled components: (i) a **retriever**, responsible for fetching relevant information from an external store based on a given query, and (ii) a **generator**, which produces the final response conditioned on both the query and the retrieved context [4, 5]. This decoupled design enhances modularity, allowing each component to be improved or evaluated independently [20].

A typical RAG pipeline consists of four sequential phases as seen in [20]:

1. **Indexing**: A knowledge corpus is curated from various sources (e.g., Wikipedia, scientific literature) and stored in a vector database. Modern implementations use dense vector embeddings generated by contrastively trained bi-encoders [11], which outperform older models in semantic similarity tasks.

2. **Searching**: Given a user query, the retriever encodes it into a dense vector and performs similarity search over the indexed document embeddings to retrieve the top-k most relevant pieces of information. To improve scalability, Approximate Nearest Neighbor (ANN) search techniques [9] are commonly used to efficiently identify close matches in high-dimensional space. Retrieved results can be further re-ranked using cross-encoder models or enhanced through hybrid retrieval methods that combine dense and sparse representations.

3. **Augmentation**: The retrieved documents are concatenated with the user query to craft a prompt, which supplies external context to the generative model. Techniques such as few-shot prompting [1] or Chain-of-Thought (CoT) prompting [18] can be applied at this stage to improve relevance, reasoning, and transparency.

   However, the effectiveness of augmentation depends not only on the quality of the prompt but also on how the context is structured and its corresponding length. Recent findings show that LLMs tend to under utilize information located in the middle of long input sequences—a positional bias known as the *"lost in the middle"* effect [7]. This highlights the importance of careful prompt formatting and strategic placement of key information to ensure it is properly attended to during generation.

4. **Generation**: A pretrained LLM produces the final response conditioned on both the original query and the retrieved context. Still, hallucinations may occur when the model fails to sufficiently rely on the provided evidence, reverting to internal knowledge shaped by pretraining biases.

Although RAG leverages *in-context learning* to enhance factuality and relevance, it does not fully eliminate challenges related to reliability, faithfulness, or reasoning accuracy. *Hallucinations* can still persist due to ineffective retrieval (incomplete, improperly ranked or noisy context), poor prompt engineering, or incomplete utilization of relevant context. These limitations highlight the need for rigorous and specialized evaluation methodologies that account for both components of the RAG pipeline and their interplay [20]. As RAG continues to be deployed in high-stakes applications, reliable evaluation becomes essential. The following section explores the necessity of RAG evaluation.

1.3 The Need for RAG Evaluation

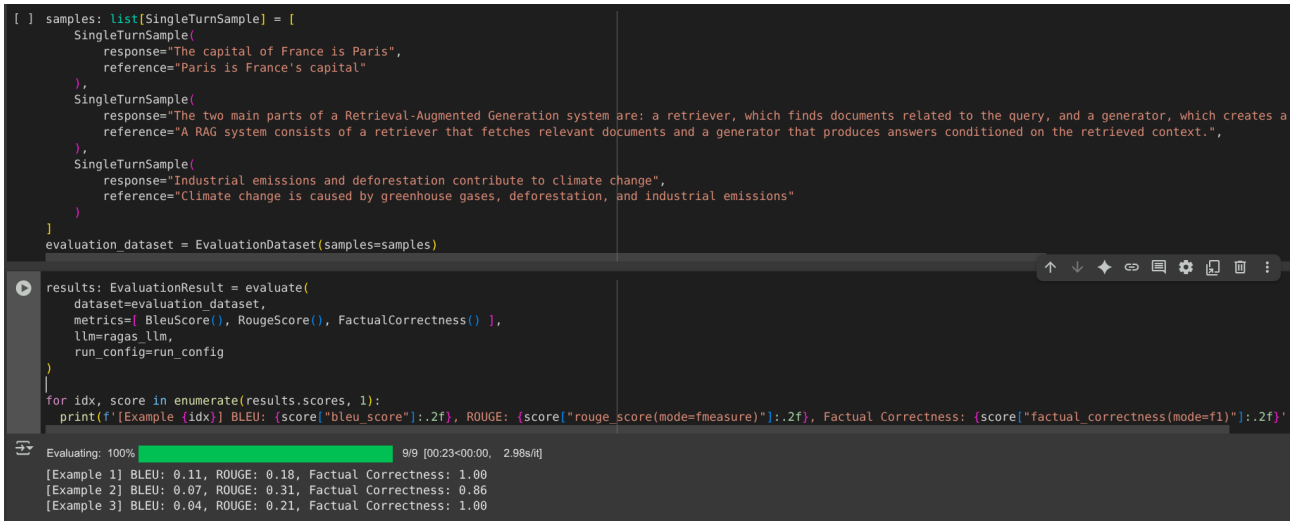
Despite the strengths of Retrieval-Augmented Generation (RAG) in addressing the core limitations of standalone LLMs—such as static knowledge, hallucinations, and non-transparency (see Table 1)—its hybrid nature introduces new challenges that necessitate rigorous and tailored evaluation strategies. A RAG pipeline comprises two loosely coupled components: a *retriever* and a *generator* [4, 5]. Each plays a distinct but interdependent role, and failures in either can degrade the overall system performance. Therefore, a comprehensive evaluation must assess the components both individually and in combination [20].

Evaluation in RAG systems occurs along two primary dimensions: *retrieval quality* and *generation quality*. Retrieval quality is often measured through *context relevance*, *precision*, and *recall*, while generation quality is mainly assessed through *faithfulness*, *correctness*, and *answer relevance* [20]. For instance, even if the retriever supplies accurate documents, the generator may still hallucinate, omit crucial facts, or produce responses misaligned with user intent. Conversely, if irrelevant or noisy passages are retrieved, even a high-quality LLM may fail to generate a relevant or factually correct response.

Traditionally, *human evaluation* has been considered the gold standard for assessing quality. Human annotators can judge nuanced aspects such as factuality, coherence, informativeness, and relevance across diverse query types. However, this process is *time-consuming*, *costly*, and *infeasible at scale*, especially when models are iteratively improved or evaluated across multiple configurations [15]. In high-stakes domains or open-ended settings, reliance on human feedback introduces a significant bottleneck. Moreover, inter-annotator agreement is not always guaranteed, and ensuring annotation consistency demands strict guidelines.

To overcome the scalability barrier of human evaluations, early RAG systems and LLMs often relied on *automated metrics* such as BLEU, ROUGE, and METEOR. These metrics compare the n-gram overlap between generated outputs and reference texts, offering quick and reproducible benchmarks. Their appeal lies in their simplicity and performance. However, these metrics are increasingly viewed as *insufficient for evaluating RAG systems*, especially those used for *question answering (Q&A)* tasks. RAG-generated answers can often be phrased in different ways but keep the semantics unchanged, which traditional metrics penalize due to lexical mismatch. Moreover, n-gram based metrics do not evaluate whether the generated output correctly integrates retrieved context, nor do they assess factual consistency [15]. Numerous studies have shown that BLEU and ROUGE *correlate poorly with human judgments*, fail to account for *semantic similarity*, and do not consider *contextual grounding*—key criteria for effective RAG output [15]. This shortcoming is illustrated in Figure 1, which contrasts the behavior of BLEU, ROUGE, and Factual Correctness on responses that are semantically equivalent but differ lexically (i.e., in their surface forms).

Figure 1: Comparison of BLEU and ROUGE with the LLM-based Factual Correctness metric from the open-source Ragas framework. Traditional n-gram metrics penalize semantically equivalent but lexically different outputs, leading to underestimation of response quality. In contrast, LLM-based metrics evaluate factual consistency and semantic alignment more robustly. This highlights the motivation for leveraging LLMs as evaluation judges in RAG pipelines.



To address these shortcomings, recent studies have introduced *LLM-as-a-judge* as a scalable, interpretable, and semantically aware alternative. In this paradigm, a powerful language model (e.g., GPT-4 [12]) is tasked with evaluating the quality of generated outputs by either assigning scores or conducting pairwise compar-

isons. This approach aligns better with human preferences and allows for *prompt customization* to target specific evaluation criteria such as *faithfulness*, *relevance*, or *correctness*. Zheng et al. [21] demonstrate that *LLM judges can achieve over 80% agreement with human raters*, effectively matching human-human agreement levels. This makes them highly suitable for evaluating complex, open-ended responses—especially in multi-turn dialogues and Q&A tasks, which is the typical setting in which RAG is used. However, this approach does have certain limitations - *position bias*, *verbosity bias*, and *self-enhancement bias* are recurring concerns. Additionally, such evaluations are *sensitive to prompt formulation*, which may lead to unstable results if not standardized across experiments.

Among the first frameworks to formalize *LLM-as-a-judge* specifically for evaluation of RAG systems is the open-source RAG evaluation framework *Ragas*[1] *(Retrieval-Augmented Generation Assessment)*. It introduces *reference-free LLM-based metrics* for evaluating critical aspects like *context relevance*, *answer relevance*, and *faithfulness* [3]. These metrics do not rely on ground-truth answers, which can help speed up the evaluation process since annotated examples are not required. Ragas provides a modular and automated framework that integrates with common RAG toolchains such as LangChain and LlamaIndex, accelerating the development and evaluation cycle for RAG-based applications [3]. This represents a significant advancement in RAG evaluation, enabling researchers and developers to perform rapid and fine-grained assessments of both retrieval and generation components.

Table 2 provides a comparative overview of existing RAG evaluation approaches, highlighting their respective strengths and trade-offs. While no single method is universally superior, the choice of evaluation strategy should be guided by the intended use case, required granularity, and available computational resources.

Table 2: Comparison of RAG Evaluation Approaches

Approach	**Advantages**	**Limitations**
Human Evaluation	Captures nuanced judgments; adaptable to diverse tasks; high validity	Costly, slow, lacks scalability; inter-annotator inconsistency
NLG Metrics (BLEU, ROUGE, METEOR)	Fast and reproducible; widely adopted; reference-based scoring	Poor correlation with human judgments; penalize semantic variation; ignore contextual grounding
LLM-as-a-Judge	Semantically aware; scalable; supports prompt customization; aligns with human preferences [21]	Sensitive to prompt design; prone to biases (e.g., verbosity, position)

Given the limitations of individual approaches, a promising direction is to combine multiple evaluation strategies. For example, using automated LLM-based metrics for scalability, supplemented by human validation for high-stakes domains like medicine, law, science, etc.

Having introduced the key limitations of standalone LLMs, presented RAG as a solution, and outlined the motivation and strategies for evaluation, the following section details the architecture and functionality of the implemented RAG application, followed by its practical evaluation using the following evaluation frameworks - Ragas, DeepEval[2] and Opik[3].

[1] https://github.com/explodinggradients/ragas
[2] https://github.com/confident-ai/deepeval
[3] https://github.com/comet-ml/opik

2 Implementation

2.1 RAG Application

To enable systematic experimentation across various configurations (see Table 3), a custom Retrieval-Augmented Generation application was developed specifically for this survey[4]. The system is fully containerized and built using the open-source R2R framework[5], with a graphical user interface powered by Streamlit[6]. It supports the core functionalities essential for RAG workflows: document ingestion, prompt- and index management, conversation tracking, and chat bot interaction.

Figure 2 depicts the system architecture, which aligns with the four key phases of a RAG pipeline discussed in Section 1.2:

- **Indexing:** Uploaded documents are parsed, chunked into semantically meaningful segments, embedded, and stored in a vector database. Users may optionally configure an index (e.g., HNSW or IVF) to improve search efficiency.
- **Retrieval:** At query time, the input is encoded into a dense vector. The top-k most similar chunks are retrieved using cosine similarity. Additionally, retrieved contexts are reranked using a cross-encoder.
- **Prompt Construction:** Retrieved context is dynamically inserted into a user-defined prompt template to construct the final input prompt.
- **Generation:** The prompt is passed to a locally hosted LLM via Ollama. The resulting response is then returned to the user.

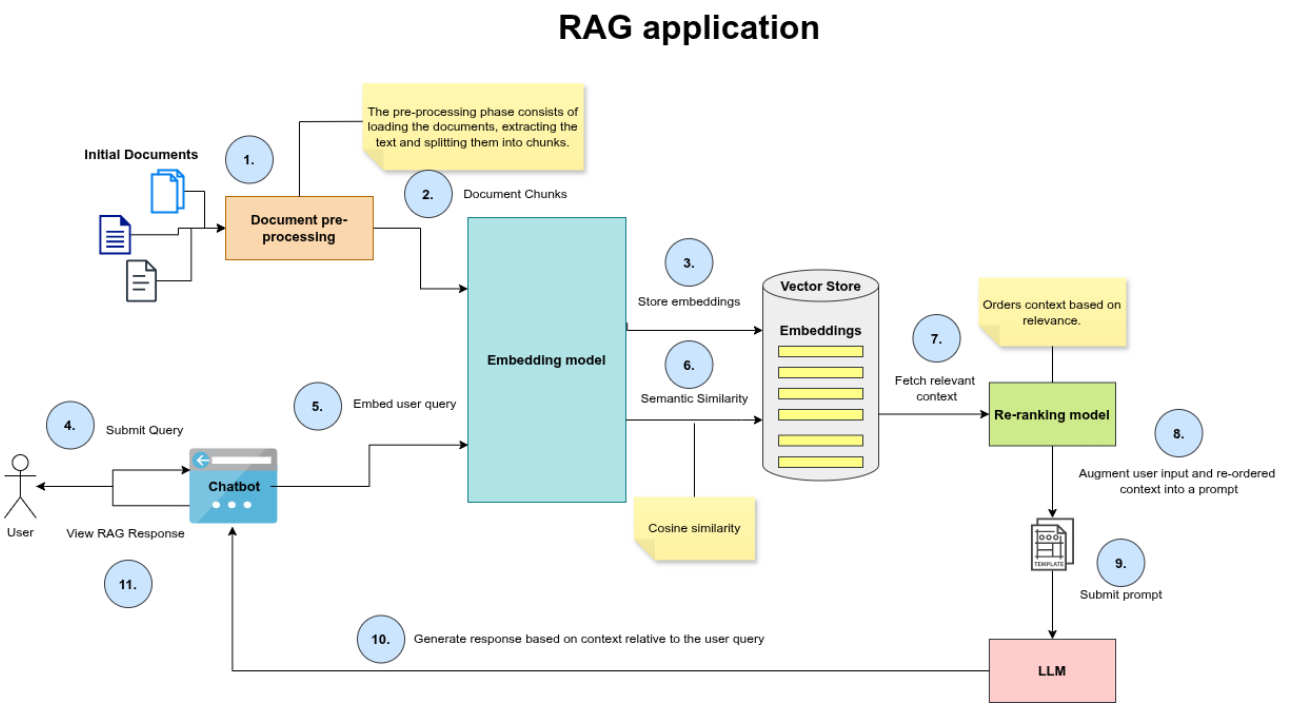


Figure 2: Architecture of the RAG application showing the four main pipeline phases: indexing (1–3), retrieval (4–7), prompt construction (8), and response generation (9–11).

This architecture facilitates flexible experimentation with retrieval parameters, LLM models, and reranking strategies via simple configuration files. For technical setup instructions, project structure, and customization options, refer to Appendix A.

[4] https://github.com/danielpetrov18/Evaluation-Approaches-for-Retrieval-Augmented-Generation-RAG-
[5] https://github.com/SciPhi-AI/R2R
[6] https://github.com/streamlit/streamlit

2.2 Experiments

To evaluate how different settings influence the RAG application's quality a series of experiments were conducted. Each experiment slightly varies one or more key parameters—such as chunk size, overlap, top-k, or the generative model. The main goal was to identify which configuration works best across different evaluation aspects like relevance, correctness, and faithfulness [20].

To ensure reliable and uniform comparison between the different settings every experiment had its own synthetic dataset, generated specifically for that configuration. This made sure that each experiment had a fair and clean setup, without any dataset bias from previous configurations. All datasets have about 50 samples and follow the same structure (see Figure 3). The table below shows the different experiment setups.

Table 3: Experiment Configurations

ID	Top-K	Chunk Size	Overlap	Model
1	3	512	0	llama3.1:8b
2	3	512	0	llama3.1:8b-instruct-q4_1
3	3	512	0	deepseek-r1:7b
4	5	768	64	llama3.1:8b
5	5	768	64	llama3.1:8b-instruct-q4_1
6	5	1024	128	llama3.1:8b
7	5	1024	128	llama3.1:8b (RAG-fusion)
8	5	1024	128	llama3.1:8b-instruct-q4_1
9	5	1024	128	deepseek-r1:7b
10	10	2048	256	llama3.1:8b

Experiments 1–3 served as the baseline with small chunk sizes and only 3 documents per retrieval. This served as a starting point to measure performance under minimal settings. Experiments 4–6 increased chunk size and overlap to see whether giving the model more context helps or introduces noise. As Liu et al. [7] points out, more context isn't always better—if chunks get too large, they can include irrelevant information or distract the LLM leading to hallucinations. Experiment 7 used an advanced retrieval method called **RAG-fusion**. Instead of relying solely on the user's original question, it generates several hypothetical variations (a technique called *query rewriting* as seen in [4]) and runs retrieval on each of them. The resulting contexts are then fused and ranked to improve the relevance of the final response [14]. This can lead to increased retrieval coverage, thus reducing missing information. However, it can also introduce redundant or noisy chunks, and it's more computationally expensive due to multiple retrieval passes. Furthermore, if the LLM fails to capture the initial user query's intent this approach can lead to even greater hallucinations than just using the basic RAG approach. Experiments 8–10 explored upper bounds by increasing chunk size and top-k to higher values. The idea was to test whether more context leads to better answers or just creates clutter. According to Lewis et al. [5], going beyond $k = 10$ on knowledge-intensive tasks doesn't significantly improve generation quality, as the increase in retrieved documents often introduces noise.

These experiments helped explore important trade-offs. Larger chunks can carry more information but might include irrelevant content, which might lead to hallucinations. Smaller chunks are cleaner but risk omitting key context. Similarly, increasing top-k boosts recall but may dilute relevance. Getting the balance right is crucial for ensuring that the model gets just enough context without being overwhelmed. Furthermore, the influence of three different generative models was observed - `lama3.1:8b`, `deepseek-r1:7b` and `llama3.1:8b-instruct-q4_1`. This set of experiments enabled the comparison of different RAG configurations and set the stage for analyzing how different evaluation metrics respond to changes.

2.3 Evaluation Datasets

Constructing high-quality evaluation datasets for thorough evaluation of RAG applications (see Figure 2) is inherently complex. Traditionally, such datasets are curated through human annotation, often involving domain experts who manually craft questions, identify relevant documents, and provide ground-truth answers. While this approach can yield highly accurate and nuanced data, it is expensive, time-consuming, and difficult to scale. Moreover, existing public Q&A datasets—such as MS MARCO or HotpotQA—are primarily designed for closed-book question answering, where answers are expected to come from the LLM's parametric memory rather than retrieved context. These datasets often suffer from poor retrieval transparency, limited query diversity, and insufficient support for evaluating grounding behavior [6].

To address these limitations, researchers have increasingly turned to Large Language Models (LLMs) as synthetic data generators. When properly guided by task-specific prompts, LLMs can generate large-scale evaluation datasets that are both customizable and reproducible [8]. In the context of RAG, such datasets are particularly valuable because they enable fine-grained control over the complexity and structure of queries and answers, allowing systematic evaluation of both retrieval and generation components.

One of the main advantages of LLM-generated datasets is scalability. Large volumes of data can be produced much faster without the need for human supervision, which is ideal for iterative experimentation across many RAG configurations. Most importantly, these datasets can be constructed with traceability in mind—that is, maintaining explicit links between the query, the retrieved context, and the reference answer—making it easier to diagnose retrieval errors separately from generation mistakes and to thus reduce the hallucinations inherent to Large Language Models. These advantages lead to a robust and task-specific data curation, which can support developers at assessing a RAG application's performance.

However, synthetic dataset generation with LLMs introduces new challenges. Without careful validation, LLMs may hallucinate facts, embed subtle factual inaccuracies that are hard to detect due to inherent biases [10] or have high costs (proprietary models like GPT-4 [12]). Furthermore, in iterative generation setups—especially those involving multi-step instruction rewriting such as Evol-Instruct [19]—there is a risk of instructional drift, where queries progressively deviate from the intended domain or complexity level. This can lead to evaluation samples that no longer reflect the original goals of the test set.

To generate reliable synthetic data for this work, `Ragas` [3] was employed using one of their datasets as a knowledge source[7]. Ragas integrates the `Evol-Instruct` framework [19], which enables the iterative evolution of base prompts into increasingly complex instructions. For the purposes of this project, two evolution strategies were selected: one that creates straightforward factual questions and another that demands reasoning over multiple information sources.

Table 4: Instruction Evolution Strategies Used in Dataset Generation

Strategy	Description	Relevance to RAG Evaluation
Simple	Generates straightforward, single-hop factual questions that can be answered using a single document chunk.	Tests basic retrieval, factual correctness and faithfulness.
Multi-Context	Produces questions requiring the combination of information across multiple chunks or documents.	Evaluates noise robustness, retrieval breadth, fusion quality, and LLM coherence across disjoint sources.

Each generated query is paired with one or more relevant context passages and a ground-truth answer (see Figure 3), allowing for isolated evaluation of retrieval- and generation quality. This structure is particularly important in modular RAG pipelines, where retrieval and generation components must be evaluated independently.

[7] https://huggingface.co/datasets/explodinggradients/ragas-airline-dataset

The synthetic datasets generated to evaluate the RAG application (see Figure 2) in this survey are designed to support a general-purpose question answering system, where most queries should be answerable from the retrieved context alone. To fulfill this goal, several properties were prioritized. First, all answers are explicitly grounded in retrieved passages to ensure alignment between context and response. Second, each dataset reflects diverse information needs by including query styles such as web-search phrasing, informal questions, long-form information requests, and comparative reasoning. Third, to ensure uniformity, consistency and reliability, all datasets were generated using the same hyperparameters-the RAG prompt template (Appendix A.4), generation model (`llama3.1:8b-instruct-q4_1`) and temperature as well as the chunking strategy used by `unstructured`[8]. Finally, each dataset entry adheres to a structured schema that includes the user input, the set of retrieved contexts, a reference answer, and the LLM-generated response.

Figure 3: Schema of synthetic datasets used in this work to evaluate a RAG application (see Figure 2). Each entry contains a query, retrieved context, ground-truth reference, and generated response.

	user_input	retrieved_contexts	response	reference
0	Ragas Airlines	[Booking Tickets Online Booking Booking a tick ...	There is no question to answer. The prompt onl ...	Special Assistance Ragas Airlines provides ser ...
1	whr is my airlin?	[Stay near the departure gate and monitor upda ...	The airline's representative will be at the ai ...	You can request assistance at least 48 hours b ...
2	What are the specific requirements for transpo ...	[Medical Equipment → Passengers may carry medi ...	Passengers may carry portable oxygen concentra ...	Passengers may carry medical devices but must ...
3	What are the key features and benefits of a go ...	[Business & First-Class Passengers → Lie-flat ...	The context highlights several key features an ...	A good customer service experience in air trav ...
4	what is airlin procedur	[Potential Issues and Resolutions for Special ...	The airline's procedure for handling special r ...	A dedicated airline staff member will escort y ...
5	airline special assistance late requests	[Potential Issues and Resolutions for Special ...	If you did not request assistance in advance, ...	If you did not request assistance in advance, ...
6	how i can chang my fligt on ragas airlines	[Managing Reservations Managing your reservati ...	To change your flight on Ragas Airlines, follo ...	To view or modify your reservation online, fol ...
7	whay do i get chrged more for my fligt if i ch ...	[2. Modification Restrictions Some tickets hav ...	If the new flight has a higher fare, you must ...	If the new flight has a higher fare, you must ...
8	What is my paymnet methode if i need to get a ...	[Check Fare Rules & Availability Review your t ...	Your original payment method will be used for ...	If your ticket is refundable, you will receive ...
9	manage my booking	[Managing Reservations Managing your reservati ...	To manage your booking, visit the Ragas Airlin ...	You can check ticket conditions and explore al ...
10	Ragas Airlines delay policy	[Flight Delays Flight delays can be caused by ...	Ragas Airlines' delay policy includes notifyin ...	When a flight is delayed, Ragas Airlines will ...
11	why cant i get email when flight is delayed	[Flight Delays Flight delays can be caused by ...	You will receive delay notifications via Email ...	You will receive delay notifications via: Emai ...
12	airline support	[Special Assistance Ragas Airlines provides sp ...	You can contact Ragas Airlines through their c ...	Depending on the length of the delay, Ragas Ai ...

2.4 Metrics

Evaluating the quality of a RAG system is inherently more complex than evaluating a standalone LLM. Unlike standard text generation tasks, RAG involves a two-step process—retrieval followed by generation—where performance depends on both the retrieval- and generation component. This complexity is further increased by the influence of the retrieved context over the generation phase which introduces the need for specialized evaluation metrics that go beyond surface-level lexical comparisons and enable component-level assessment.

Conventional metrics such as BLEU, ROUGE, and METEOR, while commonly used in natural language generation, are limited in this context. These string-based metrics are optimized for closed-book tasks and emphasize n-gram overlap with reference answers. As such, they are ill-suited for RAG pipelines where responses are grounded in external context and correctness depends on effective retrieval and faithful integration of that context into the final answer (see Figure 1). To overcome these limitations, this work adopts three LLM-based evaluation frameworks—*Ragas*, *DeepEval* and *Opik*, which follow the emerging *LLM-as-a-judge* paradigm as seen in [21]. The remainder of this chapter introduces and motivates the specific metrics selected for this study.

[8] https://github.com/Unstructured-IO/unstructured

2.4.1 Context Precision

Context Precision is a retrieval-focused metric shared across all three evaluation frameworks. It evaluates how well the top-k retrieved chunks contribute to the reference answer in terms of relevance and whether the chunks deemed relevant are ranked higher than irrelevant ones—an important factor due to the known *primacy bias* of LLMs, where higher-ranked content is more likely to be used effectively [7].

The evaluation is performed by first prompting the LLM to classify each context chunk as either `relevant` or `not relevant` based on whether it is "remotely useful" in deriving the *reference answer* relative to the *input*. This framing allows the LLM to acknowledge both direct and inferential contributions while discouraging contradictions or completely unrelated information. After each context chunk has been classified the formula below is then used to determine the final score.

High *Context Precision* scores indicate that the retriever's re-ranker is successfully surfacing pertinent knowledge in top-ranked positions—maximizing the LLM's probability to generate accurate, grounded answers. Conversely, low scores suggest poor ordering, where relevant information is buried and would probably be unused, which is more likely to cause to hallucinations or reduce answer relevance.

Mathematical Definition:

$$\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k} + \text{false positives@k}} \tag{1}$$

$$\text{ContextPrecision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Number of relevant nodes}} \tag{2}$$

Where:

- true positives@k is the sum of all relevant documents up to rank k,
- false positives@k is the sum of all irrelevant documents up to rank k,
- K is the total number of chunks in the `retrieved_contexts`,
- $v_k \in \{0, 1\}$ is the binary relevance of the chunk at rank k (relevant or irrelevant).

This formula ensures that earlier-ranked relevant chunks contribute more heavily to the final score, while irrelevant or low-ranked chunks are discounted. The result is a positional average precision normalized over all relevant chunks in the retrieved set.

Cross-Framework Comparison:

While all three frameworks share the same underlying goal of measuring how well relevant chunks are ranked within the retrieved context, they differ slightly in how this evaluation is carried out. Each framework employs a prompt that instructs the LLM to classify chunks as relevant if they are even *remotely useful* in producing the expected output.

- **Ragas** evaluates chunks individually by creating input-answer-context triplets. The LLM is asked to assess the relevance of each chunk in isolation, promoting finer-grained judgments.
- **DeepEval** takes a batched approach by submitting the input, reference answer, and a list of context chunks together, prompting the LLM to return a corresponding list of binary relevance verdicts for the entire set.
- **Opik** follows the batched approach of *DeepEval* and it introduces additional prompt constraints to filter out irrelevant, contradictory, or redundant chunks more explicitly by specifying an explicit criteria. This adds a tighter frame to its judgment process without altering the core binary decision logic.

Despite these slight differences in how context is classified, all three frameworks converge on the same binary scoring mechanism and *Weighted Context Precision* formula. As a result, they remain directly comparable in terms of both metric computation and interpretability.

2.4.2 Context Recall

Context Recall is a retrieval-focused metric available in all three evaluation frameworks. While *Context Precision* asks "What proportion of the retrieved chunks are relevant?", *Context Recall* asks "Was everything relevant retrieved". It quantifies the ratio of claims in the reference answer that can be attributed to a piece of information from the retrieved context.

This metric operates by prompting the LLM to classify individual claims in the reference (i.e., expected answer) as either `attributed` or `not attributed` based on whether the claim can be grounded in the retrieved documents.

High *Context Recall* signals that the retriever successfully fetches most of the necessary knowledge if not all, ensuring that the generative model is equipped with the factual basis needed to generate complete and grounded responses. A low score, by contrast, suggests that key information was missed entirely during retrieval—often leading to incomplete or slightly hallucinated outputs.

Mathematical Definition:

$$\text{Context Recall} = \frac{\text{Number of claims in the reference supported by the retrieved context}}{\text{Total number of claims in the reference}} \tag{3}$$

This binary claim-level classification is standardized across frameworks, allowing numerical outputs in the $[0, 1]$ and facilitating inter-framework comparison.

Cross-Framework Comparison:

Despite sharing the same objective, the frameworks differ in how they prepare and classify reference claims:

- **Ragas** and **DeepEval** treat the reference as a monolithic unit and rely on the LLM to infer which parts of it are supported by the retrieved context. The LLM is prompted to process the input, full reference answer, and top-k chunks in a single batch and classify each claim as attributable or not. This design trades off accuracy for speed and efficiency, since the LLM is prompted only once.
- **Opik** adopts a more fine-grained, two-stage approach. First, it decomposes the reference into discrete atomic statements using a specialized few-shot prompt. Then, each individual statement is independently assessed for attribution. This layered pipeline enables greater control, modularity, and transparency in verdict generation, making it better suited for explainability or tracking.

All three frameworks ultimately rely on LLM-based judgment, which introduces potential variance. To minimize this, each framework implements structured prompting with identical set of instructions and output schemas for proper data extraction.

2.4.3 Faithfulness

Faithfulness is a generation-focused metric available in all three evaluation frameworks. It quantifies the factual consistency between the generated answer and the retrieved context. It aims to penalize hallucinated content, i.e., statements that either (a) contradict the context or (b) introduce information not explicitly found within it. A faithful answer is one in which all generated claims are grounded in the retrieved context. After all claims from the response have been extracted, they get classified and the ratio of truthful claims compared to the number of all claims is the final score.

Mathematical Definition:

$$\text{Faithfulness} = \frac{\text{Number of claims consistent with the retrieved context}}{\text{Total number of claims in the response}} \tag{4}$$

Higher values indicate higher response consistency with the retrieved context, which is crucial for ensuring a hallucination-free RAG application.

Cross-Framework Comparison:

- **Ragas** adopts the following strategy: it first decomposes the generated answer into atomic, standalone factual statements and then each statement is classified as consistent or inconsistent relative to the retrieved context. The final score is then computed using the formula above.
- **DeepEval** provides a similar evaluation mechanism, although it is approached slightly differently. Both the retrieved context and response are split into atomic claims, which ensures more fine-grained classification. Thereafter, the claims from the response are checked for contradictions or introduction of information outside the context. *DeepEval* doesn't follow the binary classification model, but claims can be classified as either `consistent`, `inconsistent` or `undetermined`, where `consistent` and `undetermined` claims are grouped together. The final score is then computed using the formula above.
- **Opik** refers to this metric under the name *Hallucination* and can be interpreted as $(1 - \text{Faithfulness})$. Here, only a single prompt is submitted to the LLM and a scalar value is computed by the LLM given an evaluation criteria. The goal is to minimize the score.

Although different, all three frameworks evaluate how well the answer adheres to the retrieved context. *Ragas* and *DeepEval* follow a more structured approach, whereas *Opik* lets the LLM directly judge the faithfulness. This provides a more diverse examination.

2.4.4 Answer Relevancy

Answer Relevancy is a generation-focused metric and assesses how well a generated answer aligns with the original user input along three dimensions - completeness (fully addressing the input), conciseness (exclusion of redundant information) and relevance (matching the user's intent). Unlike factuality-focused metrics, Answer Relevancy does not evaluate whether the answer is true—only whether it is contextually appropriate and addresses the user's question meaningfully.

This metric is particularly important for evaluating the generation component of RAG systems and in particular the prompt template. Prompt engineering can significantly improve relevance by providing more explicit instructions, utilizing different prompting techniques or forcing the LLM to rely solely on the retrieved context.

Cross-Framework Comparison:

Each framework implements Answer Relevancy with a different methodology, offering complementary perspectives on response quality.

- **Ragas** uses a unique *embedding-based* approach. For a given response, the model generates N hypothetical questions that could be answered by it. These questions are then encoded into embeddings and compared via cosine similarity to the original user query's embedding as specified in the Ragas paper [3]:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^{N} \cos(E_{g_i}, E_q)$$

  Where:

  - N is the number of hypothetical questions derived from the response
  - E_{g_i} is the embedding of the i-th generated question
  - E_q is the embedding of the original question
  - cos is a function that computes the cosine similarity between the corresponding embeddings

This method gauges how semantically aligned the response is with the original input. Responses that imply or suggest content closely related to the original query will score highly, even if their surface form differs significantly. However, it may miss subtle irrelevancies due to embedding limitations.

- **DeepEval** employs an *LLM-as-a-judge* approach. It first decomposes the answer into individual statements using the LLM. Each statement is then independently classified as:
  - `relevant`,
  - `undetermined` (partially relevant or ambiguous),
  - `irrelevant`.

  The final score is computed using the following formula:

$$\text{Answer Relevancy} = \frac{\text{Number of relevant statements (relevant/undetermined)}}{\text{Total number of statements}}$$

  This method is more interpretable, allowing developers to peek behind the scenes to get a more tangible explanation of the verdicts. However, it relies heavily on the judgment of a Large Language Model and may cause inflated scores by counting statements with uncertain or weak relevance (marked as `undetermined`) as partially correct.

- **Opik** like *DeepEval* uses a LLM to determine the relevance, however it follows a more coarse-grained approach. It prompts an LLM to rate the answer on a scale from 0.0 to 1.0 along the three dimensions: **completeness**, **conciseness**, and **relevance** without decomposing it first. This approach offers a third perspective on *Answer Relevancy* by letting the LLM directly judge the pertinence given a set of evaluation criteria. However, the absence of a decomposition process makes it more subjective and opaque, which might inflate or reduce the score.

Using all three in parallel offers a broader perspective of response quality: from semantic alignment (Ragas) that is fast to compute, to fragment-level relevance (DeepEval), to holistic response evaluation (Opik). Such multifaceted evaluation is particularly important in RAG systems where generation quality is sensitive not just to relevance, but also to completeness.

2.4.5 Mean Reciprocal Rank (MRR)

(Opik only)

Mean Reciprocal Rank (MRR) is a classical metric from *information retrieval systems* that measures how effectively a retriever ranks relevant documents at the top of the result list. Specifically, it captures the position of the first relevant chunk and assigns higher scores when that chunk appears earlier in the ranking. This is particularly useful in RAG pipelines where high-ranking documents are more likely to influence the generated output due to the primacy bias [7].

Mathematical Definition:

Let:

- K be the number of queries in the dataset,
- rank_i be the rank position of the first relevant chunk for the i-th query.

Then the MRR is defined as:

$$\text{MRR} = \frac{1}{K} \sum_{i=1}^{K} \frac{1}{\text{rank}_i}$$

The score ranges from $[0, 1]$, with 1.0 indicating that the first chunk was always relevant across all queries.

In this work, the ground truth (reference answer) and each retrieved context chunk are encoded into vector embeddings using the `mxbai-embed-large` model[9]. The cosine similarity between each chunk and the reference is then calculated. The first chunk whose similarity exceeds a fixed threshold (0.75) is considered relevant, and its rank is recorded. If no chunk crosses the threshold, the rank is undefined and contributes a score of 0 to the final MRR.

This method provides a fast and deterministic way to approximate retrieval accuracy without relying on a Large Language Model. While it lacks the nuance of semantic understanding offered by LLM-judge metrics, it excels at identifying retrieval failures or degradation in ranking quality.

MRR complements *Context Precision* and *Context Recall* by emphasizing the *position* of the first useful chunk rather than observing the *ratio of relevant chunks* or their *coverage*. This makes it particularly suitable for evaluating re-ranking effectiveness in top-k retrieval scenarios.

2.4.6 Contextual Relevancy

(DeepEval only)

Contextual Relevancy is a retrieval-specific metric that evaluates the utility of each statement within the retrieved context relative to the user query. The aim is to measure how well the retrieved content aligns with what the input actually requires. Unlike broader retrieval metrics that assess overall recall or ranking, *Contextual Relevancy* operates at the granularity of individual statements. Each chunk in the retrieval context is decomposed into atomic statements, and the LLM is then prompted to judge the relevance of each statement relative to the query. This metric mainly verifies if the chunk size and top-k are optimal. This approach allows the metric to identify redundancies and irrelevant noise during the retrieval phase.

Mathematical Definition:

$$\textbf{Contextual Relevancy (CR)} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

A higher score indicates that the retriever fetches mostly or only useful information that can be utilized to answer a question properly. On the contrary, lower score signify the retrieval of irrelevant or redundant information.

2.4.7 Context Entity Recall

(Ragas only)

Context Entity Recall is a retrieval-oriented metric, which is designed to assess the effectiveness of the retriever in extracting key factual entities present in the reference answer from the vector store. Rather than evaluating the completeness of retrieved context relative to the reference, this metric focuses exclusively on named entities overlap—such as people, places, dates, or events—that are essential in heavy fact-based applications in domains like medicine, law, etc.

Mathematical Definition:

- RE be the set of entities in the reference (ground truth),
- RCE be the set of entities in the retrieved context.

$$\textbf{Context Entity Recall (CER)} = \frac{|\text{RE} \cap \text{RCE}|}{|\text{RE}|}$$

[9] https://www.mixedbread.com/docs/models/embedding/mxbai-embed-large-v1

The *Context Entity Recall* metric extracts the named entities from both the ground truth and the retrieved context via LLM prompting. To ensure consistency, multiple few-shot examples are provided to guide the LLM in recognizing and de-duplicating entity mentions (e.g., "Taj Mahal" vs. "the monument"). Once they are extracted, the number of overlapping entities across the reference and retrieved context is calculated, and the final score is derived via the formula defined above.

2.4.8 Noise Sensitivity

(Ragas only)

Noise Sensitivity measures how susceptible a RAG system is to incorporating factually incorrect information that originates from retrieved documents. This metric is designed to reveal hallucinations that arise not from the LLM itself, but from retrieval-induced noise embedded within documents that appear relevant.

In this work, *Noise Sensitivity* is evaluated in `relevant mode`, which isolates errors caused by relevant documents that include redundant content. Unlike irrelevant documents—which robust LLMs ignore—relevant but noisy passages are more dangerous because they are more likely to include misleading information, which can induce hallucinations. As such, this metric functions as a stress test for the overall quality of the RAG application.

The evaluation proceeds in three steps: First, the generated answer and ground-truth reference are decomposed into atomic factual claims using the LLM. Then, for each claim in the response the model determines its factual accuracy relative to the reference. Finally, for each factually incorrect claim in the response the metric collects all retrieved documents deemed relevant that support that claim.

Mathematical Definition:

$$\text{NS}_{\text{relevant}} = \frac{\text{Number of incorrect claims in the response supported by relevant context}}{\text{Total number of claims in the response}}$$

This definition captures the fraction of all generated claims that are both incorrect (*false positives* and *false negatives*) and inferable from relevant context—i.e., errors that result from redundant information. A higher score indicates greater vulnerability to retrieval-induced hallucinations from relative context. A lower score reflects better robustness and contextual filtering.

2.4.9 Factual Correctness

(Ragas only)

Factual Correctness evaluates the factual consistency between the generated response and reference, even if partial. This metric is particularly important in RAG applications, where generation quality must be measured not only by faithfulness or relevance, but by how faithfully it reflects the intended ground-truth content. In this survey the `F1-mode` is used to derive the final score, however `precision` and `recall` modes are also supported. Instead of using n-gram overlap, it measures factual consistency based on semantic overlap (see Figure 1). To do so the following algorithm is applied:

1. The generated answer and ground-truth reference are first decomposed into atomic factual claims via LLM prompting.
2. For each generated claim, the metric checks whether it can be supported by any of the ground-truth reference claims, if even partially.
3. Token overlap is then computed between matched claims using the `F1` mode, rewarding both high precision (relevance) and recall (completeness).

Mathematical Definition:

Factual Correctness under F1-mode is defined using the general F_β score, where $\beta = 1$:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

More generally, the F_β score is computed as:

$$F_\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

with:

- **Precision** = $\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$
- **Recall** = $\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$

In this work, $\beta = 1$ was used to give equal importance to precision and recall.

A high factual correctness score indicates that the generated answer aligns closely with the ground truth both in terms of factual content and completeness. A low score may arise due to hallucinated content, partial truths, or the omission of essential details.

Table 5: Overview of Evaluation Metrics Used in this survey. Green rows represent **retrieval-focused** metrics, yellow rows are **generation-focused**, and the red row evaluates **robustness** of the overall RAG pipeline against misleading context.

Metric	Primary Focus	Framework
Context Precision	Relevance of retrieved chunks	Ragas, DeepEval, Opik
Context Recall	Coverage of relevant answer-supporting content	Ragas, DeepEval, Opik
Faithfulness/Hallucination	Groundedness of response in context	Ragas, DeepEval, Opik
Answer Relevancy	Matching response to user intent and question scope	Ragas, DeepEval, Opik
Mean Reciprocal Rank	Rank position of first relevant chunk	Opik
Contextual Relevancy	Usefulness of each context node to the query	DeepEval
Context Entity Recall	Recall of ground-truth entities in context	Ragas
Noise Sensitivity	Robustness to relevant context containing noise	Ragas
Factual Correctness (F1)	Factual overlap between response and reference	Ragas

3 Results

3.1 Overview

This section presents the quantitative evaluation of the RAG system (see Figure 2) using the metrics introduced in the previous section (see Table 5) across ten experiments, varying chunk sizes, overlap, retrieval depth and generative models (see Table 3). The evaluation was conducted using three established frameworks: *Ragas*, *DeepEval*, and *Opik*. Table 6 summarizes all metric scores. Figures 4, 5, and 6 visually break down individual metric scores across the experiments for *Ragas*, *DeepEval* and *Opik* respectively. A simplified example of the evaluation pipeline using *Ragas* can be found under Appendix B.

Table 6: Summary of all evaluation metrics across all experiments (see Table 3) and frameworks. Best values per metric are highlighted in green, best averages per experiment in cyan. **Legend:** ↑: maximize; ↓: minimize.

Metric	Ex.1	Ex.2	Ex.3	Ex.4	Ex.5	Ex.6	Ex.7	Ex.8	Ex.9	Ex.10
Ragas										
Context Precision ↑	0.59	0.63	0.47	0.59	0.66	0.55	**0.71**	0.68	**0.71**	0.58
Context Recall ↑	0.77	0.80	0.71	**0.88**	0.82	0.83	0.87	0.84	0.87	0.85
Faithfulness ↑	0.78	0.76	0.67	**0.85**	0.77	0.77	0.79	0.81	0.66	0.76
Response Relevancy ↑	0.80	0.71	0.85	0.83	0.69	**0.94**	0.80	0.86	0.83	0.80
Context Entity Recall ↑	0.24	0.20	0.23	0.28	0.24	0.31	0.25	0.24	0.20	**0.35**
Noise Sensitivity ↓	0.24	**0.21**	0.31	0.36	0.26	0.32	0.29	0.33	0.35	0.33
Factual Correctness (F1) ↑	0.61	0.56	0.46	0.57	0.55	0.56	**0.62**	0.58	0.60	0.54
Average (Ragas)	**0.58**	**0.55**	**0.53**	**0.62**	**0.57**	**0.61**	**0.62**	**0.62**	**0.60**	**0.60**
DeepEval										
Contextual Precision ↑	0.72	**0.76**	0.63	0.65	0.69	0.65	0.67	0.67	0.67	0.56
Contextual Recall ↑	0.79	0.80	0.75	0.79	0.77	0.75	**0.81**	0.80	**0.81**	0.80
Faithfulness ↑	0.68	0.65	0.62	0.70	0.70	0.72	0.74	**0.75**	0.65	0.71
Answer Relevancy ↑	0.81	0.81	0.73	**0.84**	0.81	0.80	0.81	0.78	0.77	0.81
Contextual Relevancy ↑	0.31	**0.33**	0.31	0.29	0.30	**0.33**	0.29	0.30	0.30	0.28
Average (DeepEval)	**0.66**	**0.67**	**0.61**	**0.65**	**0.65**	**0.65**	**0.66**	**0.66**	**0.64**	**0.63**
Opik										
Context Precision ↑	0.77	**0.81**	0.75	0.71	0.71	0.71	0.72	0.71	0.72	0.62
Context Recall ↑	0.73	0.79	0.78	0.87	0.85	0.85	**0.89**	0.84	0.84	0.87
Hallucination (1-Faithfulness) ↓	0.11	0.12	0.19	**0.06**	0.13	0.12	0.15	0.11	0.11	0.21
Answer Relevance ↑	0.77	0.81	0.81	0.80	0.78	0.80	0.80	**0.85**	0.83	0.80
Mean Reciprocal Rank ↑	0.73	0.70	0.67	0.78	0.73	0.76	0.81	0.74	0.82	**0.86**
Average (Opik)	**0.62**	**0.65**	**0.64**	**0.64**	**0.64**	**0.65**	**0.67**	**0.65**	**0.66**	**0.67**

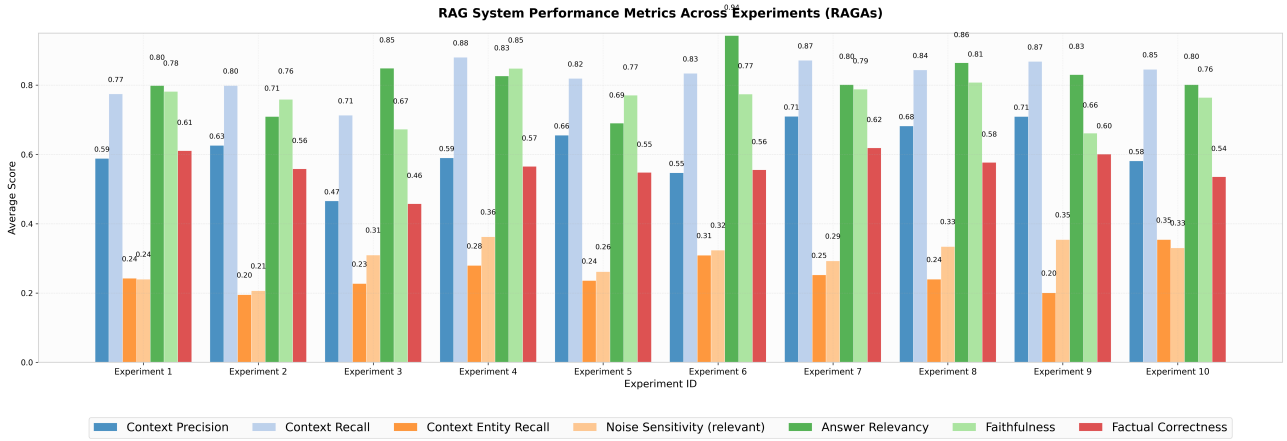


Figure 4: RAG system performance metrics across experiments (Ragas).

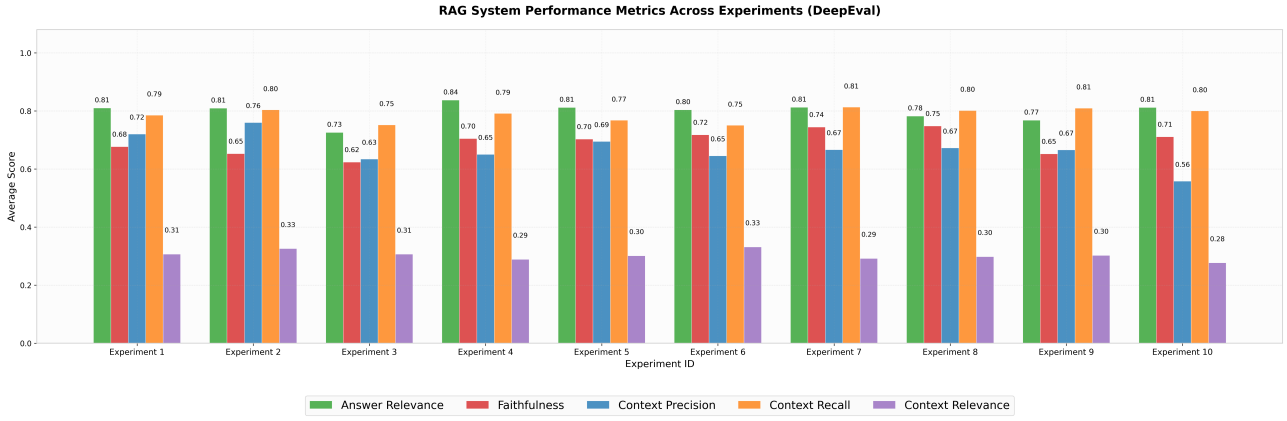


Figure 5: RAG system performance metrics across experiments (DeepEval).

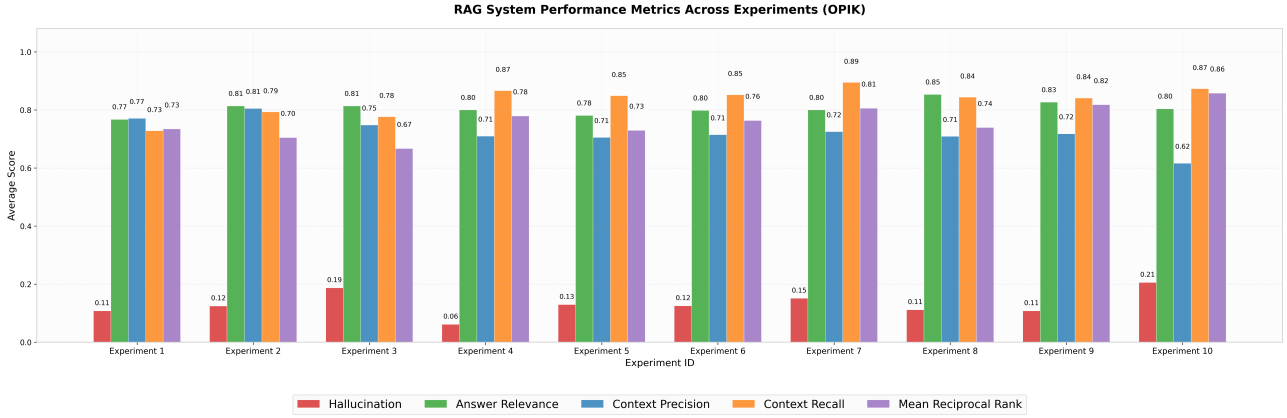


Figure 6: RAG system performance metrics across experiments (Opik).

3.2 Metrics Discussion

An interesting observation arises in the *Context Precision* metric. Both *DeepEval* and *Opik* agree that experiment 2 achieves the best result. This makes intuitive sense given the small chunk size used - 512. In contrast, *Ragas* reports the highest precision in experiments 7 and 9, where chunk sizes are twice as large,

which is completely counter-intuitive. Overall, for both *DeepEval* and *Opik* there's a clear trend - an increase in the chunk size leads to reduction in the *Context Precision* score, however for *Ragas* no such trend can be observed.

A more consistent trend emerges for *Context Recall*. All three frameworks show that increasing the retrieval hyperparameters-chunk size, overlap and top-k-improves recall, reflecting greater context coverage. Notably, *DeepEval* and *Opik* both classify experiment 7 as optimal, while *Ragas* favors experiment 4, though the recall values in *Ragas* and *Opik* are nearly identical. A key distinction between experiment 6 and 7, which adds an additional insight into recall, is the use of *RAG-Fusion* in the latter. This advanced retrieval technique leads to a noticeable improvement in *Context Recall* across all frameworks with an increase between of $4 - 6\%$, supporting the claim that RAG-Fusion can successfully lead to a higher coverage of relevant context as dicussed in [14].

Regarding the *Faithfulness* metric, both *Ragas* and *Opik* highlight experiment 4 as optimal. However, the values differ by nearly 9%, which likely reflects differences in evaluation methodology. *Ragas* employs a structured approach: it decomposes the output into atomic claims, checks their consistency with the retrieved context, and computes the ratio of grounded claims. In contrast, *Opik* relies on a single LLM call that estimates faithfulness based on holistic criteria, which may sacrifice accuracy for computational efficiency.

For *Answer Relevance*, the frameworks disagree on the best-performing experiment. This divergence is understandable given the use of different evaluation approaches, where *Ragas* relies on semantic similarity, whereas *Opik* scores the metric by prompting the LLM holistically, though the output scores still fall within a 10% range, showing some degree of consistency.

The *Noise Sensitivity* metric increases with chunk size and top-k, which aligns with the expectation that larger- and more chunks are susceptible to include more irrelevant information. However, the comparison between experiments 1 and 2 provides an additional insight: using a higher-quality LLM can increase resistance to noise. This highlights the importance of generative model quality—not just retrieval strategy—in overall RAG performance.

3.3 Summary of Observed Trends

The evaluation across all three frameworks reveals several consistent patterns. Most notably, increasing chunk size and top-k tends to enhance *Context Recall* by capturing a broader context window, yet this often comes at the expense of *Context Precision*, as retrieval becomes less focused. The trade-off between these two metrics underscores a core challenge in retrieval design: achieving high coverage without introducing irrelevant information. While larger chunk sizes and top-k values sometimes reduce *Faithfulness* due to diluted precision, configurations incorporating advanced retrieval techniques such as RAG-Fusion (e.g., experiment 7) manage to balance this trade-off effectively. Moreover, RAG-Fusion consistently improved recall across all frameworks, validating its efficacy in gathering relevant context. Differences in scoring across frameworks, especially for metrics like *Faithfulness* or *Answer Relevance*, emphasize the impact of evaluation methodology—structured and fine-granular approach vs. holistic judgment—on interpretability and reliability. Lastly, the effect of LLM quality was evident in mitigating *Noise Sensitivity*, reinforcing the importance of not only optimizing retrieval parameters but also leveraging capable generation models for faithful and context-grounded responses.

4 Conclusion

This work presents a comprehensive evaluation of a Retrieval-Augmented Generation system, emphasizing the growing role of robust and context-aware assessment methodologies. A key paradigm shift explored is the move to *LLM-as-a-judge*, where evaluation—once reliant on human annotation or limited metrics like BLEU or ROUGE—is increasingly handled by LLMs. This enables more scalable and semantically informed assessment of tasks such as question answering. The RAG system developed for this work was tested under varying configurations of chunk size, overlap, top-k, and generation models. Evaluations were conducted using three frameworks: *Ragas*, *DeepEval*, and *Opik*, each offering distinct insights through a mix of retrieval and generation metrics.

Still, the evaluation has notable limitations. The datasets were derived from a narrow knowledge base, limiting generalizability. Additionally, the experiments—while structured—were not exhaustive; a broader range of parameter combinations would yield more reliable conclusions. While metrics like *context precision*, *faithfulness*, and *answer relevance* provide valuable insights, their interpretation is sensitive to task design, LLM choice, and retrieval granularity. Continued refinement and standardization of such metrics are essential for reproducible and meaningful evaluations. In summary, this work highlights both the promise and current challenges of evaluating RAG systems with LLM-based frameworks. Future work should focus on establishing clearer metric guidelines to advance reliable RAG evaluation.

References

[1] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[2] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[3] Shahul Es et al. *Ragas: Automated Evaluation of Retrieval Augmented Generation*. 2025. arXiv: 2309.15217 [cs.CL]. URL: https://arxiv.org/abs/2309.15217.

[4] Yunfan Gao et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. arXiv: 2312.10997 [cs.CL]. URL: https://arxiv.org/abs/2312.10997.

[5] Patrick Lewis et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL]. URL: https://arxiv.org/abs/2005.11401.

[6] Rafael Teixeira de Lima et al. *Know Your RAG: Dataset Taxonomy and Generation Strategies for Evaluating RAG Systems*. 2024. arXiv: 2411.19710 [cs.IR]. URL: https://arxiv.org/abs/2411.19710.

[7] Nelson F. Liu et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. arXiv: 2307.03172 [cs.CL]. URL: https://arxiv.org/abs/2307.03172.

[8] Lin Long et al. *On LLMs-Driven Synthetic Data Generation, Curation, and Evaluation: A Survey*. 2024. arXiv: 2406.15126 [cs.CL]. URL: https://arxiv.org/abs/2406.15126.

[9] Yu. A. Malkov and D. A. Yashunin. *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*. 2018. arXiv: 1603.09320 [cs.DS]. URL: https://arxiv.org/abs/1603.09320.

[10] Nick McKenna et al. *Sources of Hallucination by Large Language Models on Inference Tasks*. 2023. arXiv: 2305.14552 [cs.CL]. URL: https://arxiv.org/abs/2305.14552.

[11] Arvind Neelakantan et al. *Text and Code Embeddings by Contrastive Pre-Training*. 2022. arXiv: 2201.10005 [cs.CL]. URL: https://arxiv.org/abs/2201.10005.

[12] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[13] Oded Ovadia et al. *Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs*. 2024. arXiv: 2312.05934 [cs.AI]. URL: https://arxiv.org/abs/2312.05934.

[14] Zackary Rackauckas. "Rag-Fusion: A New Take on Retrieval Augmented Generation". In: *International Journal on Natural Language Computing* 13.1 (Feb. 2024), pp. 37–47. ISSN: 2319-4111. DOI: 10.5121/ijnlc.2024.13103. URL: http://dx.doi.org/10.5121/ijnlc.2024.13103.

[15] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. *A Survey of Evaluation Metrics Used for NLG Systems*. 2020. arXiv: 2008.12009 [cs.CL]. URL: https://arxiv.org/abs/2008.12009.

[16] Chandan Singh et al. *Rethinking Interpretability in the Era of Large Language Models*. 2024. arXiv: 2402.01761 [cs.CL]. URL: https://arxiv.org/abs/2402.01761.

[17] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971.

[18] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[19] Can Xu et al. *WizardLM: Empowering large pre-trained language models to follow complex instructions*. 2025. arXiv: 2304.12244 [cs.CL]. URL: https://arxiv.org/abs/2304.12244.

[20] Hao Yu et al. *Evaluation of Retrieval-Augmented Generation: A Survey*. 2024. arXiv: 2405.07437 [cs.CL]. URL: https://arxiv.org/abs/2405.07437.

[21] Lianmin Zheng et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023. arXiv: 2306.05685 [cs.CL]. URL: https://arxiv.org/abs/2306.05685.

A RAG Application

A.1 Project Structure

```
docker-compose.yaml    # Defines all services
env                    # Environment variables
evaluation             # Contains evaluation notebooks
   deepeval_eval
   opik_eval
   ragas_eval          # This directory also contains a synthetic data generation notebook
experiments.csv        # Experiment configurations
img                    # Images used in notebooks and plots
project                # Source code of the app (frontend, backend)
Readme.md              # An in-depth description of the application
run.sh                 # Script to launch the RAG application with all dependencies
```

A.2 How to Access and Use the Application

To run the system locally and access its functionality, follow these steps:

1. **Disable Uncomplicated Firewall (UFW):** If UFW is active, it may interfere with container networking. You can check and disable it as follows:

   ```
   sudo systemctl status ufw
   sudo ufw disable
   ```

2. **Clone the Repository:**

   ```
   git clone https://github.com/danielpetrov18/Evaluation-Approaches-for-Retrieval-Augmented-Generation-RAG-
   cd Evaluation-Approaches-for-Retrieval-Augmented-Generation-RAG-
   ```

3. **Start the Application:** Ensure that execution permissions are set and then run the script:

   ```
   chmod u+x run.sh
   ./run.sh
   ```

   This script ensures required environment variables are exported, models are downloaded via Ollama, and all containers are launched.

4. **Verify the Deployment:**

   ```
   docker ps                              # Check if containers are running
   docker compose logs -f                 # View container logs
   curl http://localhost:7272/v3/health   # Check R2R server health
   ```

5. **Open the Web Interface:** Visit `http://localhost:8501` in your browser to access the Streamlit-based GUI.

6. **Generate and Evaluate Datasets:**

   ```
   cd evaluation
   chmod u+x setup.sh
   ./setup.sh
   ```

   This opens a JupyterLab instance where datasets can be generated using the `generate.ipynb` notebook inside `ragas_eval`.
   Evaluation notebooks are also available under `deepeval_eval`, `ragas_eval`, and `opik_eval`.
   The local Opik platform instance is accessible at `http://localhost:5173`.

7. **Shutdown and Clean-Up:**

```
docker compose down                    # Stop services
ps aux | grep ollama                   # Find Ollama process
kill <ollama_process_id>               # Kill Ollama manually
sudo ufw enable                        # Re-enable firewall if needed

# (Optional) Remove all images, containers, and volumes:
docker rmi $(docker images -q)
docker volume rm $(docker volume ls -q)
```

A.3 Customizability

- **Environment Variables:** Users can experiment with different configurations—such as chunk size, top-k, overlap, embedding model, or generation temperature—by modifying the environment variables' file at `env/rag.env`.
- **Backend Configuration:** The file `project/backend/config.toml` exposes additional low-level R2R options, including chunking strategies, embedding model's provider, etc. This allows fine-grained control over the RAG service.
- **Extending the Run Script:** The `run.sh` script can be extended to include additional setup logic (e.g., model downloads, service checks, database resets).
- **Docker Compose Extensibility:** Users can easily add new services (e.g., monitoring tools, alternative vector stores, local LLM providers) to `docker-compose.yaml`. All containers communicate via a shared network and can be exposed via ports or internal aliases.

A.4 RAG Chatbot Template

The following is a prompt template used by the chatbot to instruct the LLM on how to answer based on the retrieved context. It emphasizes factual accuracy, contextual grounding, and discourages speculation:

Listing 1: Language-agnostic prompt template used for generation

```
You are a helpful assistant in a Retrieval-Augmented Generation (RAG) system.
Your task is to answer the user's question using *only* the provided context below.

INSTRUCTIONS:
- Answer concisely and clearly using only the information from the context.
- If the context does not contain sufficient information to answer the question, state that
    clearly.
- DO NOT include citations, references, or mention the context itself.
- Do not speculate or make up information beyond what is in the context.

CONTEXT:
{context}

QUESTION:
{query}

ANSWER:
```

B Evaluation

The code snippet below provides a simplified overview of the evaluation pipeline using *Ragas*. For the complete implementation, including caching, custom prompts, and full metric configuration, refer to the full notebook available here. Additionally, for *DeepEval* and for *Opik*.

B.1 Ragas

```
from ragas.metrics import (
    LLMContextPrecisionWithReference, LLMContextRecall,
    ContextEntityRecall, NoiseSensitivity, ResponseRelevancy,
    Faithfulness, FactualCorrectness
)
from ragas.evaluation import evaluate

# Define metrics
context_precision = LLMContextPrecisionWithReference(...)
context_recall = LLMContextRecall(...)
# ... other metrics ...

# Run evaluation
results = evaluate(
    dataset=evaluation_dataset,
    metrics=[
        context_precision, context_recall, context_entity_recall,
        noise_sensitivity, response_relevancy, faithfulness,
        factual_correctness
    ],
    llm=ragas_llm,
    embeddings=ragas_embeddings
)
```