Explainable Large Language Models for RAG Systems in Financial Applications

Ibrahim Al Khalil RIDENE `ibrahim-al-khalil.ridene@student-cs.fr`
Kerrian Le Caillec `kerrian.lecaillec@student-cs.fr`
Lydia Hammache `lydia.hammache@student-cs.fr`

April 2025

Abstract

Large language models (LLMs) have achieved remarkable performance across diverse natural language processing tasks, yet their opaque decision-making processes pose significant challenges for trust, accountability, and practical deployment in high-stakes domains. Explainable Artificial Intelligence (XAI) methods, developed in the case of LLMs, aim to address these challenges by making their outputs more interpretable, their reasoning transparent, and their biases identifiable. This paper explores state-of-the-art explainability techniques for black box LLMs, including counterfactual explanations. We further examine the integration of explainability into retrieval-augmented generation (RAG) frameworks, which provide verifiable evidence for generated content. The proposed approaches not only enhance model transparency but also improve the reliability in applications requiring rigorous validation as it the case in finance. Through qualitative and quantitative evaluation, we demonstrate the efficacy of these techniques in improving the interpretability of LLM outputs. This work provides a foundation for the development of robust, explainable LLM systems.

Keywords: Language models, Explainable AI

Code of the project accessible on: **GitHub**

Contents

1 Introduction

Large Language Models (LLMs) such as GPT [15] and LLaMA have become foundational tools in modern natural language processing, excelling in tasks like question answering, summarization, and reasoning. Despite their impressive capabilities, LLMs are often considered "black-box" systems due to their lack of interpretability and traceability. This poses a significant challenge in critical domains like finance, where model transparency, factual grounding, and trustworthiness are essential.

To mitigate these concerns, Retrieval-Augmented Generation (RAG) [11] has emerged as a promising architecture. A RAG system enhances the generation process by incorporating relevant, external information retrieved from a corpus. Specifically, given a user query q, the RAG pipeline first retrieves a set of supporting documents c_k grouped in a context defined as $\mathcal{C} := \{c_1, \dots, c_k\}$ based on similarity measures (e.g., cosine similarity), and then uses them alongside the query to generate a grounded response y:

$$p(y \mid q) = \sum_{c \in \mathcal{C}(q)} p(y \mid q, c) p(c \mid q).$$

This document-retrieval mechanism provides interpretability by linking the model's answer to specific, verifiable evidence, making RAG systems inherently more explainable than traditional LLMs. The two primary components of a RAG system are:

- **Retrieval Module:** Selects the top-k most relevant documents from a large external corpus based on semantic similarity to the query.
- **Generation Module:** Generates a final response conditioned on the query and retrieved documents.

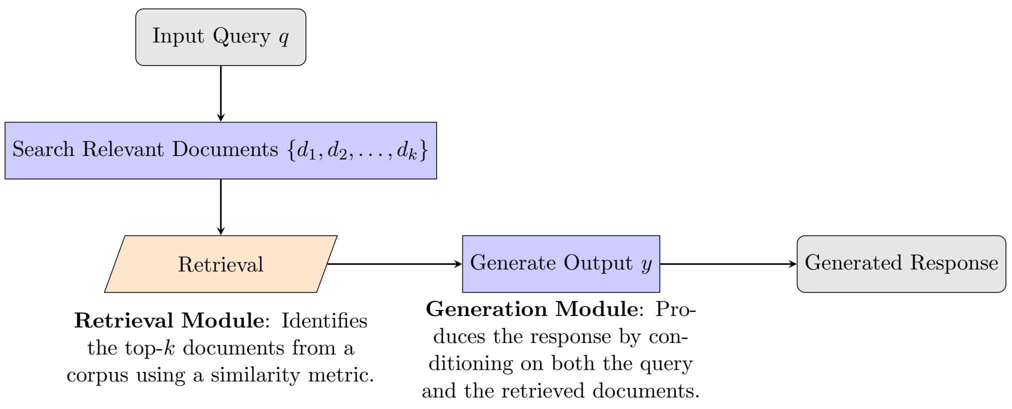

Input Query q
Search Relevant Documents $\{d_1, d_2, \dots, d_k\}$
Retrieval
Generate Output y
Generated Response
Retrieval Module: Identifies the top-k documents from a corpus using a similarity metric.
Generation Module: Produces the response by conditioning on both the query and the retrieved documents.

Figure 1: Illustration of a RAG (Retrieval-Augmented Generation) architecture.

Explainable Artificial Intelligence (XAI) refers to a set of methods aimed at making AI model predictions more understandable and interpretable by humans. When applied to LLMs and RAG systems, XAI helps trace back how decisions are made, identify influential inputs, and detect potentially harmful biases or hallucinations. In this context, explainability is achieved through techniques such as attribution methods (quantifying input contributions), ablation testing (removing parts of context to test sensitivity), rationale extraction (isolating influential context spans), and hallucination detection (verifying if output is grounded in retrieved data).

Our goal is to combine these techniques to evaluate and explain the end-to-end decision-making of a RAG system in the financial domain. This is particularly important when dealing with financial documents like reports, where accuracy and transparency are necessary.

2 State of the art

Many frameworks have been developed with the idea of explainability of language models in mind. Techniques have modified traditional explainable methods to sentiment analysis and other classification tasks like LIME [16], which approximate an estimator using a simpler model around a certain point in the output space, to language modelling [12]. The main problem with usual explainability techniques is their lack of adaptation to non-quantitative tasks. Nonetheless, applications of Shapley coefficients, which quantify the importance of a feature compared to the overall models, have been used to evaluate the changes in text generation [8].

Other approaches related to large language models uses the underlying transformer architecture, notably the attention mechanism in order to understand how tokens interact [3]. We can also consider techniques which tries to detect potential issues with an output based solely on the feature space of the underlying transformers.

In this report, we will focus on more modern approaches to explainability explores the use of language models as estimators for different metrics, we can cite GroUSE [13] or RAGAS [5] among others. These frameworks rely on standard language models like GPT models and ground-truth labels in order to evaluate and establish benchmarks for different criterion. They were specifically designed for RAG models. These methods have the advantage of being compatible with black-box models i.e. the parameters of the network are unknown as well as precisely curated for text generation tasks.

Other framework have emerged with the motivation of mitigating the toxicity biases that can exist in model's outputs [1].

3 Methodology

In this project, we design and analyze a Retrieval-Augmented Generation (RAG) system with a strong focus on transparency and explainability. Our approach is divided into two primary axes: retrieval explainability, which aims to answer ”why such documents have been retrieved?” and generation explainability which aims to answer ”why did the model answer in this manner?”.

First, we study the retrieval stage by comparing various document retrievers. These include models trained on general-purpose corpora and financial-domain data. We examine their ability to return relevant and informative documents, which form the foundation of the generation phase.

Secondly, we focus on the generation step. Given a set of retrieved documents, we use a large language model (LLM) to answer finance-related questions. To make this process explainable, we analyze how each document contributes to the final answer, test the robustness of the generated outputs through ablation, and employ a second LLM as a judge to detect hallucinations or unsupported claims.

Each component of the RAG pipeline is implemented with explainability in mind. This allows us to not only measure performance, but also to trace and justify the reasoning behind each generated response.

4 Retrieval Evaluation

The retrieval module plays a foundational role in Retrieval-Augmented Generation (RAG) systems. Its ability to accurately surface relevant and contextually rich information directly influences the quality, factuality, and trustworthiness of the generated responses. In the context of financial applications—where precision is paramount—evaluating the retrieval performance is crucial to ensure that the downstream generation step is grounded in the most informative evidence.

This section presents a detailed assessment of different retrieval strategies, focusing on the effectiveness of various models, embedding techniques, and similarity search methods. By comparing these methods, we aim to identify configurations that yield the most reliable document selection for financial question answering.

4.1 Dataset

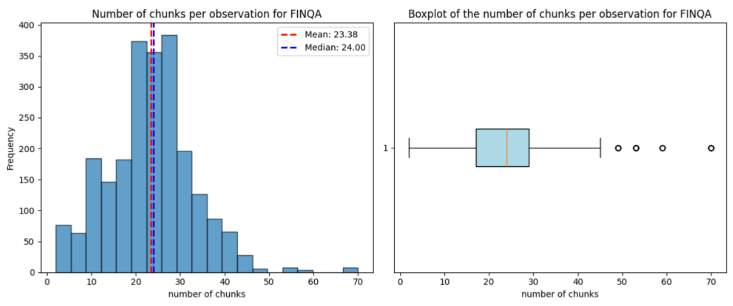
To assess the retrieval effectiveness in a controlled and explainable setting, we use the **RAG-Bench** benchmark [6, 7], which provides question-answering datasets across various domains. Each example includes a query, a list of candidate chunks, and expert-annotated relevant chunks—enabling fine-grained evaluation of retrieval performance.

We focus on the **FinQA** and **TAT-QA** subsets, which are tailored for financial applications. As summarized in Table 1, FinQA emphasizes deep reasoning over financial reports, including both textual and tabular data, and typically requires high financial literacy. In contrast, TAT-QA involves broader domains and focuses on complex numerical reasoning over semi-structured tables.

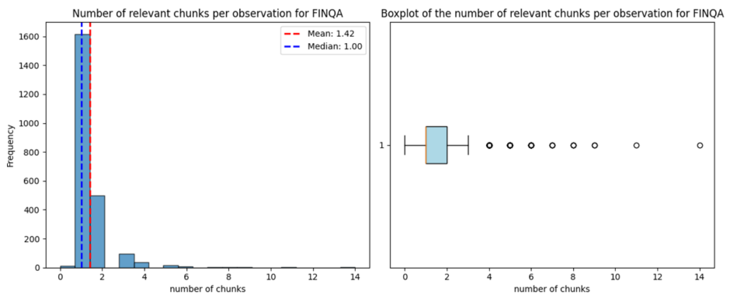
Feature	**FinQA**	**TAT-QA**
Main Focus	Financial reports and reasoning	Tabular data across multiple domains
Data Type	Text + Tables (Financial context)	Tables (Semi-structured, diverse topics)
Reasoning Complexity	High financial literacy required	Strong numerical reasoning across tables
Use Case	Financial document Q&A	General tabular data understanding

Table 1: Key differences between FinQA and TAT-QA.

To better understand dataset structure, we visualize the distribution of total chunks and relevant chunks per observation. For **FinQA** (Fig. 2), each question is accompanied by an average of 23 chunks, with typically only 1–2 being relevant. This supports the need for effective top-k retrieval. Choosing $k = 5$ ensures that relevant context is likely captured without introducing excessive noise, since we can capture approximatively all the relevant chunks defined in the botton plot with this value.



(a) Total chunks per observation.



(b) Relevant chunks per observation.

Figure 2: FinQA: Distribution of total (a) and relevant (b) chunks per observation.

Similarly, for **TAT-QA** (Figure 3), the average number of chunks is lower (around 9), and again, the number of relevant ones remains close to 1. Thus, setting $k = 5$ remains a suitable compromise for capturing useful context while avoiding irrelevant information.

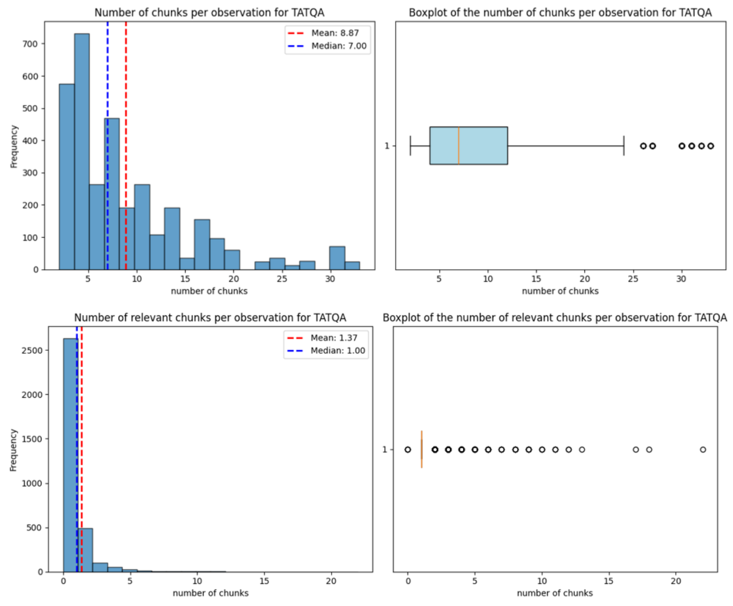


Figure 3: TAT-QA: Distribution of total (top) and relevant (bottom) chunks per observation.

4.2 Retrieval Model Selection

4.2.1 Models

We evaluate four retrieval models, each leveraging different training strategies and domain specializations to produce high-quality text embeddings. Below is a summary of their key characteristics:

- **E5** [17]: A model trained with weak supervision using contrastive learning. It is optimized for zero-shot retrieval tasks, where positive pairs are selected based on top-ranked search engine results and negative pairs from low-ranked results.
- **Contriever** [10]: A retrieval model trained using unsupervised contrastive learning. It builds sentence embeddings from adjacent text segments as positive pairs and random segments as negative ones. Its domain-agnostic nature allows general-purpose retrieval.
- **FinBERT** [2]: A BERT-based model fine-tuned on large-scale financial corpora. It has shown strong performance on finance-specific tasks such as sentiment analysis, named entity recognition, and classification.
- **FinGPT** [9]: A domain-specific language model tailored for financial tasks, including question answering, summarization of reports, and retrieval-augmented generation. It integrates a retrieval component into its fine-tuning process for better factual grounding.

4.2.2 Metrics

To evaluate the performance of retrieval models, we consider standard information retrieval metrics that quantify how effectively relevant documents are ranked and selected among the top-k retrieved chunks. The following metrics are used:

- **Recall@K:** Measures the proportion of relevant chunks successfully retrieved among the ground-truth relevant chunks.

$$\text{Recall@K} = \frac{\text{Number of Relevant Retrieved Chunks in Top } K}{\text{Total Number of Ground-Truth Relevant Chunks}}$$

- **NDCG@K (Normalized Discounted Cumulative Gain):** Evaluates the ranking quality of the retrieved results by assigning higher scores to relevant documents that appear earlier in the ranked list.

$$\text{DCG}_K = \sum_{i=1}^{K} \frac{\text{Relevance}_i}{\log_2(i+1)}, \quad \text{NDCG}_K = \frac{\text{DCG}_K}{\text{Ideal DCG}_K}$$

  This metric captures both the presence of relevant documents and their position in the top-k results, providing a more nuanced evaluation.

Although **Precision@K** is a commonly used metric, we discard it in our evaluation. Precision is defined as:

$$\text{Precision@K} = \frac{\text{Number of Relevant Retrieved Chunks in Top } K}{K}$$

However, its denominator is fixed at K, making it highly sensitive to the choice of K rather than model behavior. In contrast, recall and NDCG are more robust and better reflect retrieval quality in our setting.

4.2.3 Results

To evaluate the retrieval quality of different embedding models, we conduct experiments using the RAGBench benchmark on both the **FinQA** and **TAT-QA** datasets. Each retriever (E5, Contriever, FinBERT, and FinGPT) is tasked with returning the top-k relevant chunks per question. Performance is assessed using two metrics: **Recall@k** and **NDCG@k**, which evaluate relevance coverage and ranking quality respectively.

Figure 6 presents the evaluation results. On the FinQA dataset, E5 and Contriever show significantly higher recall and NDCG compared to FinBERT and FinGPT. A similar trend appears in the TAT-QA dataset, though FinBERT and FinGPT perform comparatively better there than on FinQA. These results highlight the robustness of E5 and Contriever across financial and semi-structured tabular domains.

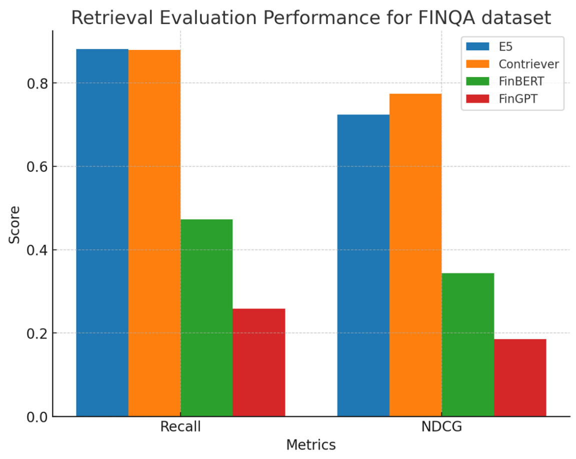


Figure 4: Retrieval performance on FinQA dataset using Recall and NDCG.

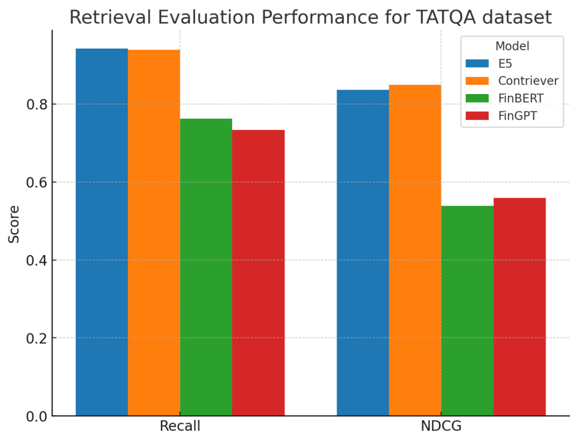


Figure 5: Retrieval performance on TAT-QA dataset using Recall and NDCG.

Figure 6: Comparison of retrieval performance across models.

While E5 leads in recall, it performs slightly worse than Contriever in terms of NDCG, indicating less effective ranking of the relevant documents. It is also significantly larger (560M parameters) and slower in inference. FinBERT and FinGPT, although domain-specific, are not tailored for retrieval tasks. FinBERT being optimized for sentiment classification and FinGPT primarily for generative financial tasks. Contriever, however, strikes a practical balance: it is fast (110M parameters), highly competitive in performance, and was trained using an unsupervised contrastive learning approach that generalizes well across tasks. Based on these factors, we select Contriever as our retrieval model of choice.

4.3 Embedding Type

After selecting Contriever as our retrieval model, we evaluate how different embedding aggregation strategies impact its performance. Specifically, we compare:

- **Mean Embedding**: The average over all token embeddings in the output.
- **Sum Embedding**: The sum of all token embeddings.
- **CLS Token Embedding**: The embedding corresponding to the special classification token [CLS].

Figure 7 shows retrieval performance on both FinQA and TAT-QA datasets using these strategies. We observe that the **mean and sum embeddings yield comparable performance**, significantly outperforming the CLS embedding in both Recall and NDCG.

This similarity is expected, as both mean and sum pooling aggregate information across the entire sequence and provide a comprehensive representation of the input. While sum embeddings are scale-dependent (i.e., sensitive to input length), the retrieval evaluation here suggests this did not negatively affect performance, likely due to consistent chunk sizes.

In contrast, CLS embeddings perform poorly. This is likely because Contriever, as a contrastive retriever, was not trained to rely on the [CLS] token for semantic representation. In transformer architectures like BERT, the [CLS] token is typically used for classification tasks, where a condensed representation of the entire input is needed. However, for retrieval tasks—especially those involving dense similarity comparisons across large corpora—aggregating all token-level representations (as in mean or sum pooling) leads to more robust and informative embeddings.

As a result, **we retain the mean embedding strategy** for its simplicity, length-invariant behavior, and strong performance.

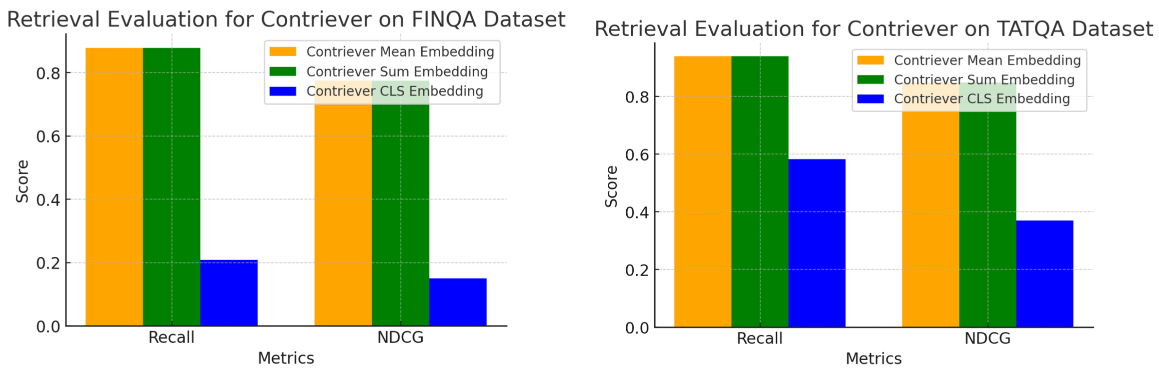


Figure 7: Retrieval performance of Contriever with different embedding types on FinQA (left) and TAT-QA (right).

4.4 Search Method

In this step, we compare two document retrieval strategies for the selected Contriever model: standard cosine similarity (which computes the distances between the query and all the chunks and find the closest k chunks to the query) and FAISS (Facebook AI Similarity Search), an approximate nearest neighbor (ANN) method.

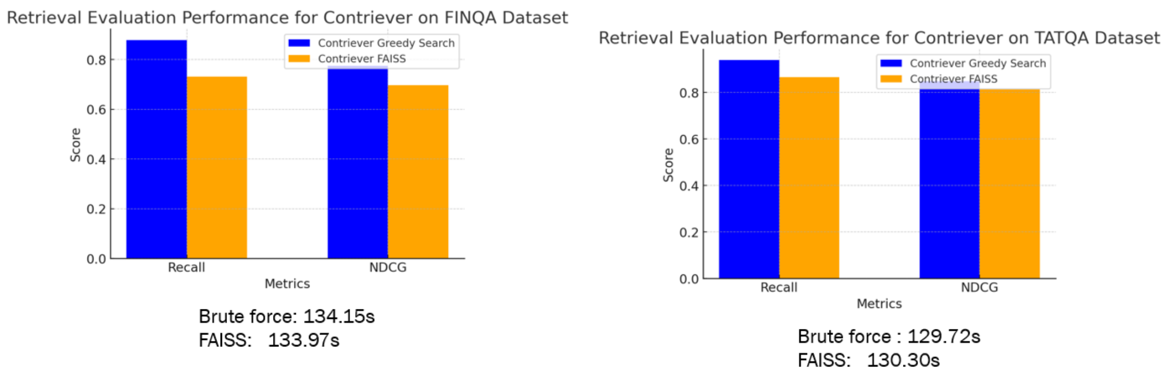


Figure 8: Comparison between brute force and FAISS on FinQA (left) and TAT-QA (right) datasets.

As shown in Figure 8, standard search slightly outperforms FAISS in both Recall and NDCG across the FinQA and TAT-QA datasets. In terms of runtime, both approaches are nearly equivalent (around 130 seconds) due to the limited number of chunks per observation in the test datasets.

Feature	Cosine Similarity (Standard search)	FAISS (ANN Search)
Accuracy	Exact	Approximate
Speed	Slow for large datasets	Fast (sublinear)
Scalability	Poor for >10K docs	Excellent >1M docs)
Memory Usage	High (stores all embeddings)	Efficient (uses indexing)

Table 2: Comparison of brute force vs FAISS for retrieval.

Although FAISS is more scalable and memory-efficient, brute standard search offers better retrieval quality due to its exact similarity computation. Since our retrieval task operates on a relatively small number of documents per query (see Section 4.1), the scalability and indexing advantages of FAISS do not yield practical improvements here.

Given its superior performance and negligible difference in runtime under our conditions, we adopt the **brute force search strategy** for all subsequent experiments.

4.5 Cosine vs. SVM Retrieval

In previous retrieval steps, we primarily relied on **cosine similarity** to rank document chunks based on their semantic closeness to the query. While this method is effective for capturing direct relevance, it does not explicitly promote *diversity* among the retrieved documents. As a result, multiple top-k chunks may be semantically redundant or clustered around similar content.

Cosine Similarity. Computes a score purely based on angular distance between the query embedding and each chunk embedding. It selects top-k chunks with highest cosine scores:

$$\text{cosine\_sim}(q, d_i) = \frac{\langle q, d_i \rangle}{\|q\| \|d_i\|}$$

This favors highly similar chunks, which may be redundant. To address this limitation, we introduce an alternative approach using a **Linear SVM classifier** as a reranking mechanism. This classifier is trained with a single positive example (the query embedding) and multiple negatives (chunk embeddings), with the goal of identifying diverse yet relevant passages. The underlying idea is illustrated in Figure 9.

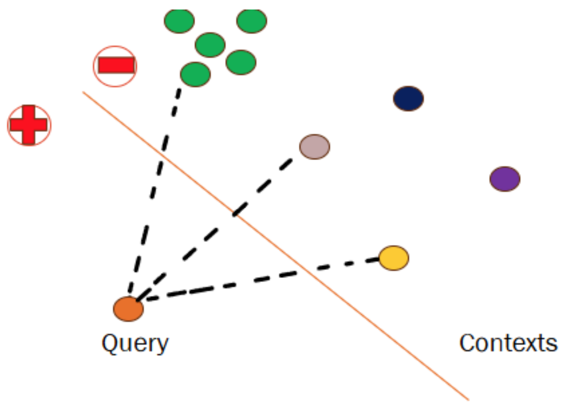


Figure 9: Illustration of SVM-based retrieval. SVM selects chunks near the decision boundary to balance similarity and diversity. The redistribution of context clusters (very similar chunks) around the decision boundary satisfy this diversity

SVM-Based Retrieval. Trains a linear classifier with:

- **Positive class**: the query embedding.
- **Negative class**: all chunk embeddings.

After training, documents with the smallest absolute margin from the SVM decision boundary are selected. These represent chunks that are close to the query but also spread across the feature space—balancing **similarity and diversity**.

In summary, while cosine similarity ensures relevance, SVM introduces a diversity-aware perspective by selecting informative boundary instances. In downstream generation tasks, this can improve factual grounding and reduce redundancy in retrieved contexts.

5 Generation Evaluation

After retrieving relevant chunks in the previous stage, the generation step takes as input both the **user query** and the corresponding **retrieved context** to synthesize a coherent and informative answer. This step plays a critical role in grounding the response in evidence and minimizing factual inconsistencies.

In this section, we evaluate the quality and reliability of the generated outputs, particularly focusing on their factual consistency and explainability.

Our evaluation is twofold. First, we use **ablation testing** to measure the sensitivity of the generation to each retrieved context, analyzing which passages are most influential in forming the final answer. In a second time, we apply a **large language model as a judge** to perform both *hallucination detection* and *rationale extraction.* This combined strategy helps us understand how generation depends on the input and assess its alignment with the underlying context.

5.1 Ablation Testing

In this subsection, we aim to assess the influence of each retrieved context chunk on the final generated answer. This is achieved through an **ablation testing** strategy, where we iteratively remove one context chunk at a time and regenerate the response using the same query. By comparing the resulting output to the original full-context response, we estimate how essential each chunk was in shaping the answer.

Let the original question be denoted by q, and the retrieved set of context chunks as $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$. The original response r is generated by a language model as:

$$r = \mathrm{LLM}(q, \mathcal{C})$$

For each $i \in \{1, \ldots, k\}$, we remove chunk c_i to create a reduced context $\mathcal{C}_{-i} = \mathcal{C} \setminus \{c_i\}$ and generate a new response $r_{-i} = \mathrm{LLM}(q, \mathcal{C}_{-i})$.

To quantify the influence of c_i, we compute the semantic similarity $\mathrm{sim}(r, r_{-i})$ between the full-context response r and the reduced-context response r_{-i}. This similarity is computed using cosine similarity between their sentence embeddings:

$$\mathrm{cosine\_sim}(r, r_{-i}) = \frac{\langle \phi(r), \phi(r_{-i}) \rangle}{\|\phi(r)\| \|\phi(r_{-i})\|}$$

where $\phi(\cdot)$ denotes the sentence embedding function (here, provided by SBERT).

Chunks that, when removed, cause a significant drop in similarity are deemed more important. This method offers a fine-grained view of context sensitivity and helps to identify which retrieved pieces of information are most influential in shaping the model's response.

5.2 LLM-as-a-Judge

To assess the factual correctness and trustworthiness of generated responses, we adopt a technique where a second Large Language Model (LLM) is employed as an external *judge*. A LLM-as-a-judge serves as an evaluator for defining and refining metrics based on another language model data. It also helps identify which context elements most strongly support the answer (rationale extraction).

This method introduces an additional layer of validation into the RAG pipeline, as shown in Figure 10, reinforcing explainability and factual consistency.

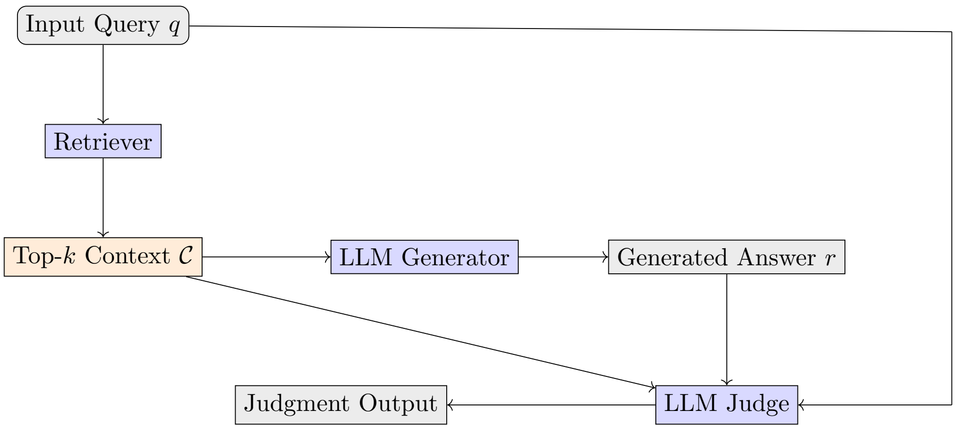


Figure 10: LLM as a Judge integrated within a RAG pipeline to evaluate factual consistency.

5.2.1 Hallucination Detection

In this part, we leverage a judge LLM to assess whether the generated answer includes hallucinated content not grounded in the retrieved context.

To evaluate hallucination detection capabilities of our LLM-based judge, we use the **RAGTruth** dataset [14]. This benchmark provides human-annotated labels for hallucinations in generation tasks, including *question answering*, *summarization*, and *data-to-text generation* (see Appendix 8.1). Each sample includes the input query, the generated response, the associated reference context, and labeled spans identifying two types of hallucinations: *conflicts* and *baseless information*.

RAGTruth is especially well-suited to our objective because it not only offers diverse generation settings, but also includes fine-grained annotations of hallucinated spans, allowing us to test both binary hallucination detection and the accuracy of span-level identification. This makes it a reliable benchmark for assessing both classification and explainability performance of LLM judges.

Method We use the RAGTruth dataset to evaluate the hallucination detection capabilities of various LLMs acting as factuality judges. The process is structured in three stages:

1. **Prompt Construction:** Each RAGTruth instance provides a writing prompt (e.g., a question, structured data, or passage), which serves as the query. It is paired with relevant context and a generated answer from the base LLM.

2. **Judgment by LLM:** The constructed input (query + reference + generated output) is passed to a candidate judge LLM (e.g., LLaMA 3.3-70B Instruct). The LLM is prompted to identify hallucinated spans using a JSON-based template.

3. **Comparison with Ground Truth:** The LLM output is compared to the human-labeled hallucinations provided in RAGTruth. Evaluation metrics such as precision, recall, and F1 score are used to assess judgment accuracy.

This setup enables us to systematically compare different LLMs for their ability to serve as reliable, explainable hallucination detectors in downstream applications like QA, summarization, and data-to-text generation.

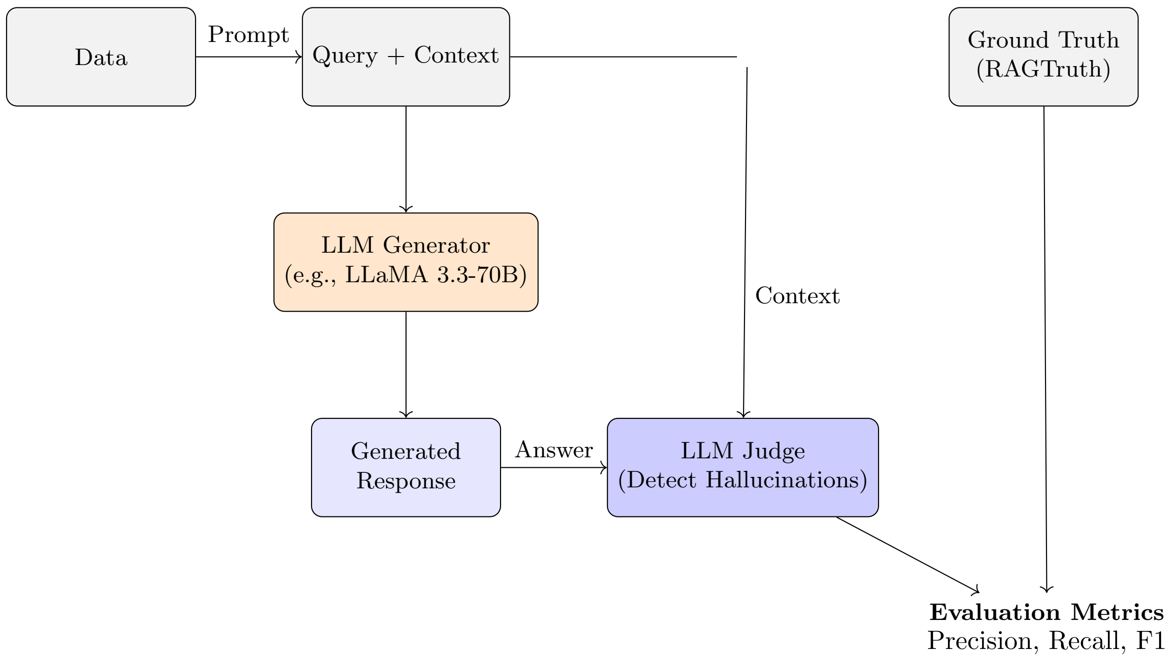
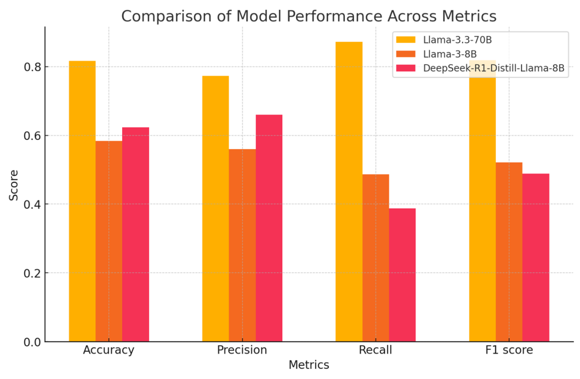
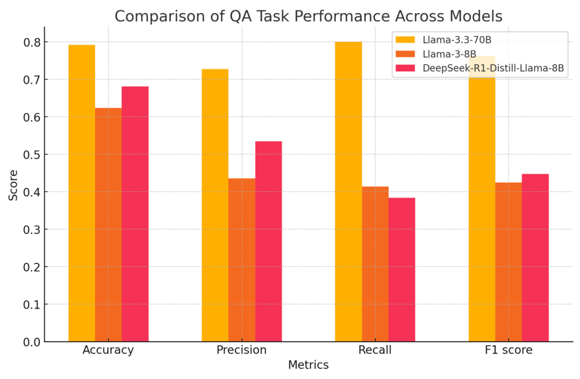


Figure 11: RAGTruth-based pipeline for evaluating LLMs as hallucination judges.

Results We evaluate the hallucination detection performance of three different LLMs: **Llama-3.3-70B**, **Llama-3-8B**, and **DeepSeek-R1-Distill-Llama-8B**. The evaluation is performed both globally across all task types (QA, Summarization, Data2Text), and specifically on the QA subset, which is our main focus for financial data.



(a) All Tasks (QA, Summary, Data2Text)



(b) QA Task Only

Figure 12: Comparison of hallucination detection models across tasks.

Figure 12 displays the aggregated metrics. Llama-3.3-70B clearly outperforms the other models across most metrics, especially in Recall and F1 score. In the QA-specific plot, the

Llama-3.3-70B continues to show dominant performance, confirming its robustness in detecting hallucinations in question-answering contexts.

Given that hallucination detection is a binary classification problem with a class imbalance (as shown in Appendix 8.1), we prioritize the **F1 score** as the primary selection metric. This ensures balanced sensitivity and precision.

We thus select **Llama-3.3-70B** as our hallucination judge. Despite being a large and computationally expensive model, its superior performance is crucial in our context—*financial applications cannot tolerate hallucinations due to the high-stakes nature of decision-making.*

For better visibility on numerical performance of each judge, refer to Appendix 8.2.

5.2.2 Rationale Extraction

In this part, we focus on enhancing the transparency of the RAG system by extracting the **rationale**, i.e., the specific parts of the retrieved context that directly support the generated answer using the LLM as a judge. While hallucination detection verifies the correctness of generated answers, rationale extraction *explains* them.

Given a query q, a generated answer a, and a set of retrieved context chunks $\mathcal{C} = \{c_1, c_2, ..., c_k\}$, the goal is to extract a subset $\mathcal{R} \subseteq \mathcal{C}$ such that $\mathcal{R}$ justifies the answer a. Mathematically, this can be seen as:

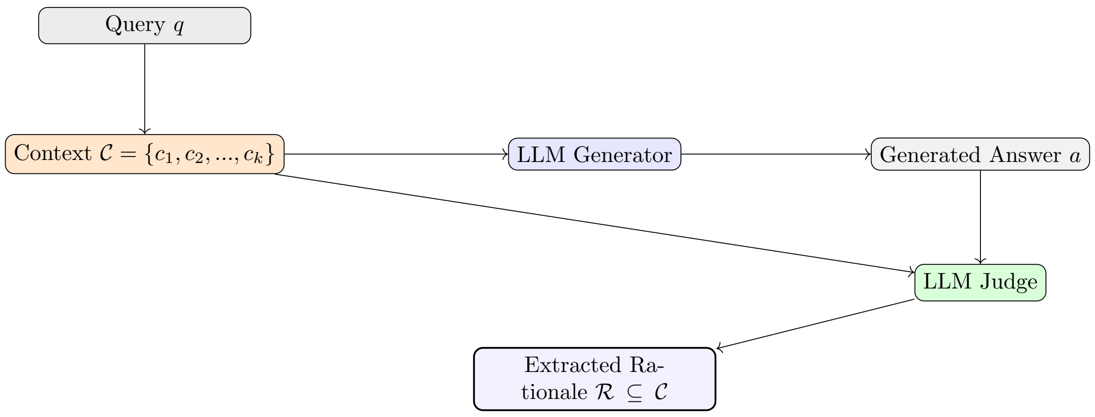
$$\mathcal{R} = \arg\max_{\mathcal{S} \subseteq \mathcal{C}} \text{support}(a \mid \mathcal{S})$$

where support$(a \mid \mathcal{S})$ represents how well subset $\mathcal{S}$ of context chunks explains or grounds the answer a. In practice, we approximate this by prompting a powerful LLM (LLaMA-3.3-70B) with the full context and instructing it to output only the key supporting segments. The prompt used explicitly forbids summarization or explanation, enforcing strict adherence to evidence.

Prompt-based Extraction:

The system constructs a prompt structured as follows:

- It includes the original *Question*, *Context* (concatenation of all retrieved passages), and the *Generated Answer*.
- It asks the model to output a list of bullet points containing only the exact spans from the context that justify the answer.
- Any additional reasoning, rewording, or summarization is explicitly forbidden.



Query q
Context $\mathcal{C} = \{c_1, c_2, ..., c_k\}$
LLM Generator
Generated Answer a
LLM Judge
Extracted Rationale $\mathcal{R} \subseteq \mathcal{C}$

Figure 13: Rationale extraction process: relevant parts of the context are selected to support the generated answer.

This is particularly useful for end-users or analysts seeking transparency on why the model produced a specific answer. It is also an important step toward building trust in AI-driven systems, especially in sensitive domains like finance.

5.2.3 Metric rationales

Faithfulness

Faithfulness measures how accurately an LLM-generated output aligns with the provided context. Using the LLM-as-a-Judge, faithfulness can be quantified by asking the model to:

- Identify the number of true statements T in the output.
- Compare these to the total number of statements S.

This yields a faithfulness metric defined as:

$$\text{Faithfulness} = \frac{T}{S}.$$

The LLM evaluates T by cross-referencing statements with the context and assigning truth values. The notion of "statement" can be a bit vague in the sense that it is up to the model to decide whether a sentence in the answer a should be split in several parts or whether it should stay as a unique statement. This obviously can alter the metric precision as the value can change based on the potential splits done by a model. If we can assess robustness in the metric definition for a single LLM-as-a-judge model, we have no guarantees that between two models the same metric value will be returned.

Answer Relevancy (AR)

Answer relevancy evaluates how well the generated response answers the original query. Using the LLM-as-a-Judge, the following process is employed:

- The query q is used to generate sub-queries q_i from the provided output.
- The similarity between q_i and q is computed using cosine similarity.

The final AR score is computed as:

$$\text{AR} = \frac{1}{n}\sum_{i=1}^{n} \text{cosine\_sim}(q_i, q).$$

Completeness

Completeness assesses whether the output captures all relevant information from the context. The LLM assigns a grade (e.g., on a scale from 1 to 5) by evaluating how well the information in the output covers the key points in the context. Other similar metric exist and do not rely on a grade system for their valuation as it is the case with the following metric. One way to measure the completeness of a prompt is to compute what is called the context relevancy. Context relevancy measures the proportion of sentences in the output that are directly relevant to the provided context. Formally:

$$\text{CR} = \frac{\#\text{relevant sentences}}{\#\text{total sentences}}.$$

The LLM is used to identify and count relevant sentences based on their alignment with the context.

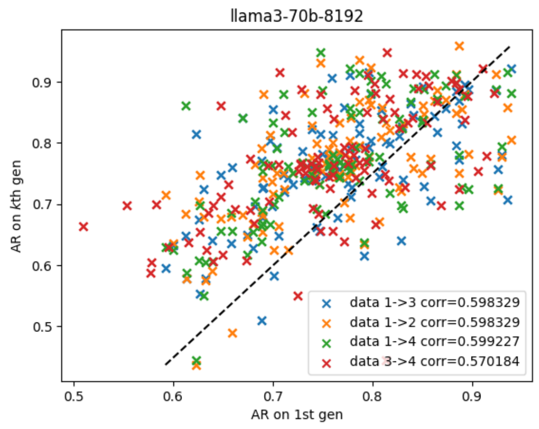
Positive Acceptance and Negative Rejection

In scenarios where the model must decide whether to respond, LLM-as-a-Judge evaluates:

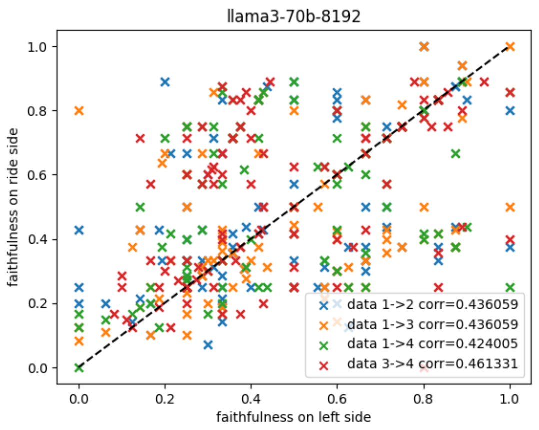
- Positive Acceptance: Did the model respond appropriately when it should have?
- Negative Rejection: Did the model refrain from responding when it was not supposed to?

These metrics ensure the model's decision boundaries align with user expectations. Furthermore, as none of the metric models rely on a ground-truth for their evaluation, we can use them in inference, without having to rely on pre-conceived examples to asses the quality of the RAG model.

Nonetheless, this leads to a new uncertainty in the metric evaluation, as we have no guarantees that the LLM-as-a-judge output will be identical over a range of models. While this may be expected in a machine learning framework, what is more problematic is the non-robustness of the output for the same model. This is a problem when considering relatively high values of temperature T for the LLM-as-a-judge model. The temperature acts in the softmax selection of tokens by the transformer model, the probability of a token being chosen by the model is inversely proportionate to T, meaning that when $T \to 0$, the model will always pick the token with the higher probability, while $T \to \infty$ will lead to a selection of the tokens based on a uniform law over the whole vocabulary. This impacts all models including large models like `LLaMa-3-70B`, for instance if we take $T = 1$ for the same input a LLM-as-a-judge can return drastically different results, as illustrated in Fig. 14. The figure depicts for the same inputs $\mathcal{X} = \{(q_j, \mathcal{C}_j, a_j)\}_{j=1}^N$, different values of faithfulness and answer relevancy, as previously defined. We scatter plotted the pairs of values of the metric for different iterations i and i' i.e. $(\text{Faithfulness}^{(i)}(\mathcal{X}), \text{Faithfulness}^{(i')}(\mathcal{X}))$. In an optimal setting, all the point should lie on the identity line depicted in black. As we can see, this is not the case as the pairs of metric values are scatter around the space. Therefore, choosing values of temperature around $T = 0.1$ should reduce this issue.



(a) Answer relevancy scattered for several iterations of the LLM-as-a-judge model.



(b) Faithfulness.

Figure 14: Illustration of the effect of the temperature T on the robustness of the metrics.

Another solution to the robustness issue is to consider a way to average multiple iterations of the judge model for the same input. The value should in fact be more accurate with respect to the input.

5.3 Factor-based classification

An issue related to only using a straight LLM-as-a-judge model is that the model has no guarantees of actually giving a confident measure for a specific task. Furthermore, with several conditions to respect for a metric evaluation, the LLM may struggle in verifying each condition before submitting a metric value. In fact, the problem could be seen as establishing a confidence metric C, which evaluates the reliability of the generated outputs for a given criterion. For instance, C could be equal to 1 if the generated prompt is hallucinated and 0 otherwise. To evaluate such a confidence metric, we would use a supervised classifier $\hat{f}_\theta$, which would take several elementary metric values m_j, that we will refer as the factors of the model. The estimated classifier $\hat{f}_\theta$, which can be implemented using feedforward neural networks (FFNN) or other machine learning models, is trained to assign a value of C given the input which is a set of factors $\mathbf{m} = (m_j)_{j=1}^k$. This decision-making process relies on the conditional probability $f(\mathbf{m}) = P(C = 1|\mathbf{m})$. The supervised nature of the classifier ensures the learnable nature of C, whereas previously, with the LLM-as-a-judge model alone, we could not assess the trust we had in an output. The factor-based classifier is then another step on top of the judge model.

The training data comprises pairs of evaluation metrics and corresponding confidence labels, $\{(\mathbf{m}^{(i)}, C^{(i)})\}_{i=1}^N$, sampled from the input-output pairs of the model. To generate the factors, we average over a few iterations the output of the LLM-as-a-judge model to ensure robust estimates of the factors. This statistical foundation ensures that the classifier can generalize effectively across the dataset, even when annotations are limited. In such scenarios, few-shot learning methods are employed, allowing the model to extrapolate from small labeled datasets.

Hallucination detection is treated as an outlier identification problem within the joint distribution of factors $\mathbf{m}$ and confidence labels C. This approach requires robust statistical techniques to ensure reliable performance across diverse settings. Annotated datasets play a crucial role in this process, providing the necessary ground truth for model training. Using this setting, one can easily use statistical properties of supervised models. However, the limited availability of such data creates challenges that are addressed through advanced sampling and estimation techniques. In addition, parameters derived from a dataset with generic prompt are optimized not only for in-distribution settings but also in new examples, ensuring that the model maintains adaptability and reliability. Indeed, we have no idea that the classifier $\hat{f}_\theta$ trained on a particular dataset will replicate the same results on another dataset. Nonetheless, the criteria we use are general in the sense they can be applied without much difference to more specific datasets.

In the Table 3, we propose an application for hallucination detection using the RAGTruth dataset. We use the $\mathbf{m}$ =(faithfulness, answer relevancy, context relevancy) metric set as the input of the classifier model, and the confidence metric C is equal to 1 if the RAG output a is hallucinated. Using the RAGTruth dataset, we can evaluate the results of the classifier using traditional evaluation metric for supervised classification. We considered several types of classifier but a simple feed-forward neural network (FFNN) with two hidden layers manage to achieve an accuracy which is defined as Accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$, slightly better than the benchmark model which is just a LLM-as-judge outputting an hallucination score in $\{0, 1\}$. We also computed the correlation factor between the classifier's output and the ground-truth as this metric is sometimes preferred to better grasp the overall trends of both the ground-truth and the classifier's output. The benchmark is provided by the results given in the Appendix 8.2.

Classifier	Judge temperature	Accuracy	Correlation factor
FFNN	1	0.66	0.32
	0.1	**0.81**	**0.60**
Gradient Boosting	1	0.53	0.01
	0.1	0.61	0.30
Benchmark	0.1	0.79	0.58

Table 3: Results of the different methods for the RAGTruth

Another point interesting to mention with a probabilistic approach to the whole classifier model is the use of statistical tool in context without labels usually in production to ensure the stability of the model. An interesting opening on this matter can be found in [4].

6 Explainability

In the previous sections, we mainly explore rationale extraction for both the retrieval and generation parts of the RAG to evaluate an answer given several elements. Nonetheless, as this may give us an indication of how reliable a specific query and its response can be. With an adequate artillery of metrics, we are able to better understand the overall reaction of the RAG but understanding why it failed is not obvious. Indeed, this approach is not enough to assess the whole explainability of the overall structure. Several methods exist to assess explainability in the realm of language models.

6.1 Chain-of-thought prompting

Developed in Wei et al. [18], Chain of Thought (CoT) reasoning improves explainability in RAG systems by structuring the generative process into discrete and interpretable components. CoT introduces an intermediate reasoning layer that explicitly links retrieved documents to inference steps. Specifically, CoT decomposes the model's output into a series of logical or mathematical steps, when dealing with computations. Based on the retrieved evidence, the model is able to explain the steps it takes. This process allows for systematic attribution of the generated response to specific retrievals, making it possible to evaluate not just the correctness of the final output but also the coherence and relevance of intermediate reasoning. By enforcing such structure, CoT reduces the opaqueness of the generated answer and transforms the generation process into a more modular pipeline that can be independently validated and debugged. To implement CoT reasoning, the easiest technique is to impose a ”meta-prompt” that will ask the generation model of the RAG to do CoT reasoning when answering the user's query given the context.

6.2 Judge explanation

Based on the previous construction of a CoT reasoning, one can apply the same logic to a LLM-as-a-judge and give an explanation of the whole evaluation by extracting specific statements, notably for hallucination detection the judge model can extract the hallucinated part. This entails a better understanding of both models; on the one hand, we can better understand the decision of the judge based on the provided justification and we can also understand what causes issues with the RAG model for a specific query.

To evaluate the justifications provided, we use the RAGTruth dataset which provides specific sequence considered as hallucinated by the annotators. One can compare the two statements with ordinary natural language processing metrics such as ROUGE or BLEU score which uses

n-grams (contiguous sequence of n words):

$$\text{BLEU} = e^{1-r/c} \exp\left(\sum_{n=1}^{N} \frac{\log p_n}{N} \right),$$

where:

- c is the length of the generated text, and r is the length of the reference text.
- p_n is the n-gram precision, calculated as:

$$p_n = \frac{\text{Number of matching } n\text{-grams}}{\text{Total number of } n\text{-grams in the generated text}}.$$

- N is the maximum n-gram size (commonly $N = 4$).

$$\text{ROUGE-}N = \frac{\sum_{S \in \text{References}} \sum_{n\text{-gram} \in S} \min(\text{Count}_n(\text{Generated}), \text{Count}_n(S))}{\sum_{S \in \text{References}} \sum_{n\text{-gram} \in S} \text{Count}_n(S)},$$

where:

- S contiguous subset of the reference text
- Count_n: The number of occurrences of a specific n-gram in the input text.

The scores measures how similar two sequence are to each other by counting how many words are identical. While this is a really simple way to measure similarity between two statements, it is useful in the context of assessing that the retrieved statement is close to the annotated one. The scores are equal to 1 when the two sequences match exactly and 0 if there is no shared n-grams. To evaluate the judge explanation, we compare the hallucination detected by the judge with the annotated sequence, we have plotted the BLEU vs. ROUGE scores on the Fig. 15. What wee see is that the majority of scores are not null which indicates that the detected sequences corresponds to the hallucination for most of the cases. Some discrepancies exist, notably on the inclusion of small parts of the sequence either in the retrieved segment or the annotated one which are not shared, thus the scores are not equal to one. The takeaway message is that the judge model manages overall to detect the hallucinated sequence in a RAG answer.

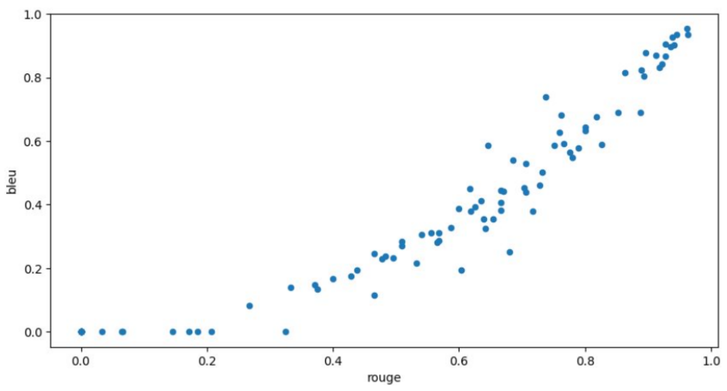


Figure 15: BLEU vs. ROUGE scores for the detected hallucinations in the RAGTruth dataset

6.3 Feature attribution

In the previous sections, we studied explainability methods tailored for natural language processing, nonetheless it may be also interesting to study more conventional XAI methods in order to provide a more systematic approach to explaining potential issues with a RAG generated answers. One way to do this is to apply a similar approach to the one presented in [8], which introduces the TokenSHAP algorithm. If we look at the embedding of the output of a user query q and a context $\mathcal{C}$, one can easily remove tokens or words from the concatenated prompt and evaluate how different the two modified embedding is from the reference one b. The idea to measure the importance of a specific word i is to remove it from the prompt, compute the embedding v_{-i} and evaluate the $c_{-i} = \text{cosine\_sim}(b, v_{-i})$. The importance of the word i is then defined as $imp_i = \frac{1}{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} c_{-j} - c_{-i}$ (we could refer to this quantity as the Shapley value of the word as well), N being the length of the prompt. We can easily do statistics with the importance quantity over different queries, as this can highlight if specific terms regularly appear in hallucination cases or other issues with the model. This could indicate a problem with unknown words. This particular method relies on the intrinsic qualities of an embedding notably the invariance between similar meaning. An example of possible statistics has been done on the TAT-QA dataset and is illustrated in Fig. 16. We plotted the average of the importance quantity of a specific word over different hallucinated prompts. We used the embedding of a LLaMa-3-1.7B to establish the statistics as it is a fairly light model. The results shows the cases where the RAG struggled to generate an adequate answer. A flaw of this approach is that terms like dates or quantities will have a really high importance as they are the ones with high impacts on the generated approach. Nonetheless, the approach is useful in detecting words causing problems to the RAG model.

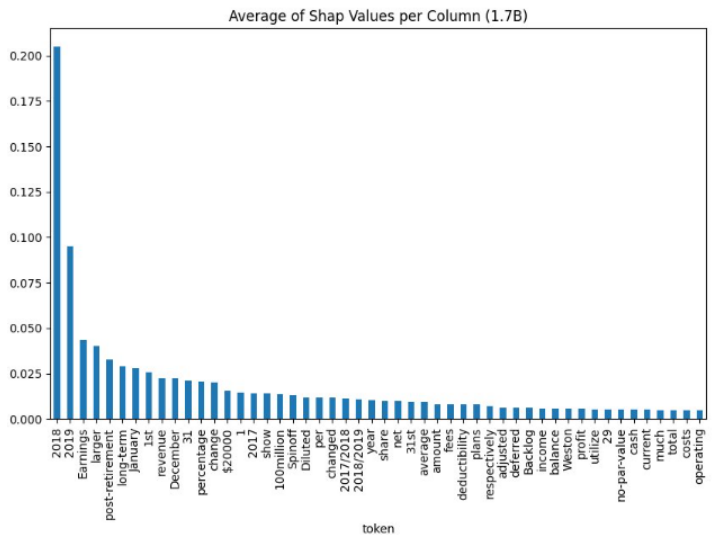


Figure 16: Average importance values for detected hallucinations in the TAT-QA dataset

7 Pipeline

In this section, we present the full pipeline of our explainable RAG-based system tailored for financial document question answering. Building upon the components previously studied in isolation, we now integrate them into a complete, end-to-end architecture that emphasizes transparency and factual correctness.

This section bridges the gap between research experimentation and real-world deployment, showing how explainability can be embedded directly into an AI pipeline for reliable financial decision support.

7.1 Architecture

Figure 17 illustrates the overall architecture of our production-grade RAG system. It integrates document ingestion, information retrieval, generation, and judgment into a single explainable pipeline tailored for financial applications.

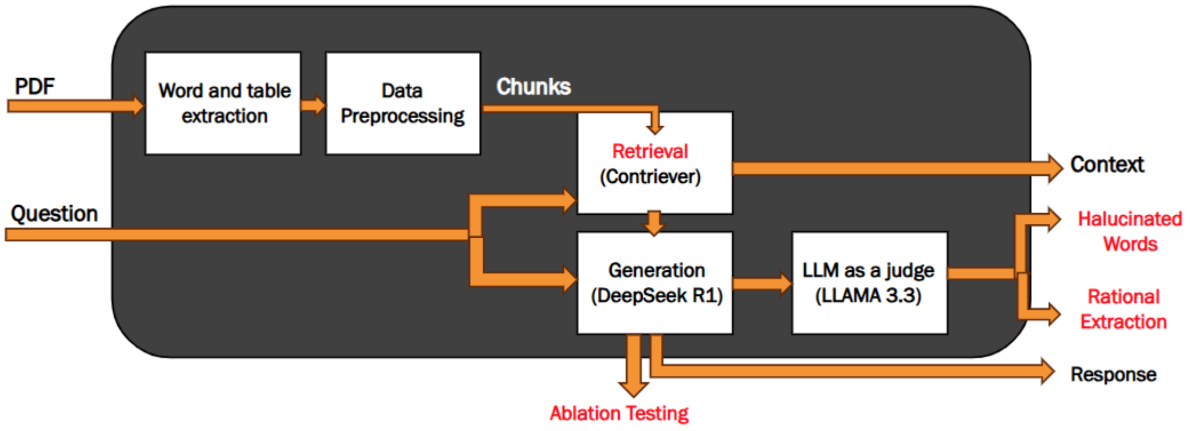


Figure 17: End-to-end explainable RAG pipeline for financial question answering.

The pipeline operates as follows:

- **Document Ingestion:** Financial reports in PDF format are first processed to extract raw text and tables using structured parsing tools.
- **Data Preprocessing:** The extracted content is segmented into manageable chunks to facilitate retrieval.
- **Retrieval:** Given a user query, the *Contriever* model selects the most semantically relevant document chunks from the preprocessed input. The explainability in this step is achieved by examining the similarity scores (e.g., cosine similarity) between the query and each chunk, enabling us to trace why a particular chunk was selected.
- **Generation:** The retrieved context, along with the query, is passed to the *DeepSeek R1* generator to produce an informed answer. The generator also supports ablation testing to trace influence of individual chunks, and outputs rationale spans and hallucinated phrases for transparency.
- **LLM as a Judge:** The generated answer is verified by a second LLM (*LLaMA-3.3-70B*), which flags hallucinated content and highlights rationales — providing an interpretable judgment layer.

Notably, components highlighted in red in the figure denote where explainability is enforced: context similarity scores during retrieval, ablation-based impact tracking, hallucination detection, and rationale extraction. This setup ensures that not only is the answer grounded in factual content, but also that the decision-making process is traceable and auditable — critical for high-stakes domains like finance.

7.2 Practical Examples

We evaluated our pipeline implementation on a concise financial report to closely analyze the results and assess the architecture's capacity to provide explainability. The corresponding examples are presented in Appendix 8.3.

8 Appendix

8.1 RAGTruth dataset

The RAGTruth dataset is commonly used to evaluate hallucination in Retrieval-Augmented Generation (RAG) systems. It includes multiple task types such as data-to-text generation, question answering (QA), and summarization. The table below shows the distribution of hallucinated ('Label = 1') and non-hallucinated ('Label = 0') examples across different tasks.

Task Type	**Hallucination Label**	**Count**
Data2txt	0 (No Hallucination)	90
Data2txt	1 (Hallucination)	210
QA	0 (No Hallucination)	196
QA	1 (Hallucination)	99
Summary	0 (No Hallucination)	194
Summary	1 (Hallucination)	106

Table 4: Class imbalance in the hallucination labels across task types.

8.2 LLM-as-a-Judge Performance on RAGTruth Dataset

The tables below show the performance of three different large language models (LLMs) used as judges for hallucination detection across three tasks (QA, Summary, Data2Text) on the RAGTruth dataset. The models vary in size and architecture: LLaMA-3 70B, LLaMA-3 8B, and DeepSeek-R1-Distill-LLaMA-8B.

Table 5: LLaMA-3 70B Performance on RAGTruth Dataset

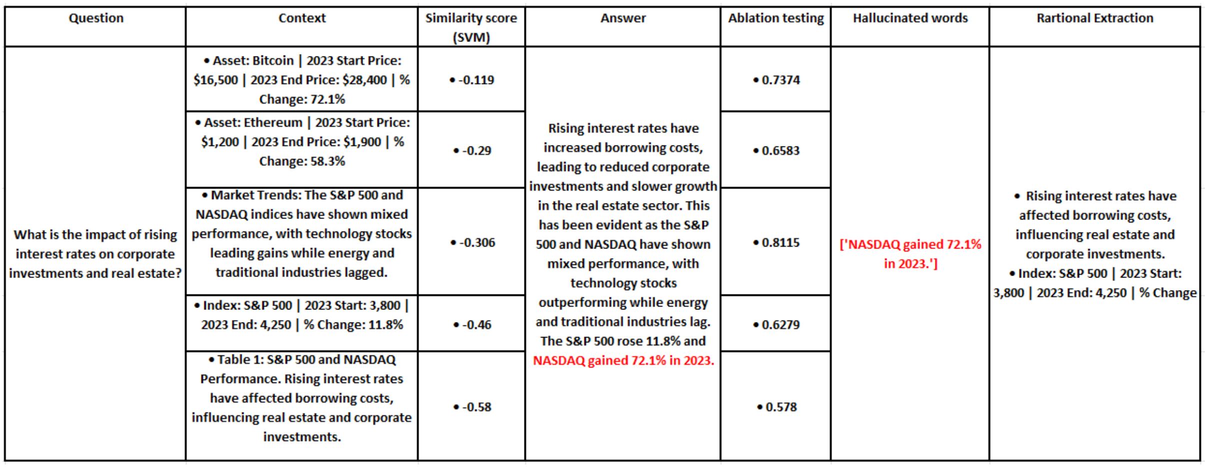
Task	**Nb samples**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Overall performance	895	0.817	0.773	0.872	0.819
QA	295	0.792	0.727	0.800	0.762
Summary	300	0.815	0.786	0.846	0.815
Data2Text	300	0.839	0.789	0.938	0.857

Table 6: LLaMA-3 8B Performance on RAGTruth Dataset

Task	**Nb samples**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Overall performance	895	0.584	0.560	0.487	0.521
QA	295	0.624	0.436	0.414	0.425
Summary	300	0.583	0.398	0.349	0.372
Data2Text	300	0.547	0.713	0.590	0.646

Table 7: DeepSeek-R1-Distill-LLaMA-8B Performance on RAGTruth Dataset

Task	**Nb samples**	**Accuracy**	**Precision**	**Recall**	**F1 score**
Overall performance	895	0.623	0.660	0.388	0.489
QA	295	0.681	0.535	0.384	0.447
Summary	300	0.593	0.367	0.208	0.265
Data2Text	300	0.597	0.894	0.481	0.625



Question	Context	Similarity score (SVM)	Answer	Ablation testing	Hallucinated words	Rartional Extraction
What is the impact of rising interest rates on corporate investments and real estate?	• Asset: Bitcoin | 2023 Start Price: $16,500 | 2023 End Price: $28,400 | % Change: 72.1%	• -0.119	Rising interest rates have increased borrowing costs, leading to reduced corporate investments and slower growth in the real estate sector. This has been evident as the S&P 500 and NASDAQ have shown mixed performance, with technology stocks outperforming while energy and traditional industries lag. The S&P 500 rose 11.8% and NASDAQ gained 72.1% in 2023.	• 0.7374	['NASDAQ gained 72.1% in 2023.']	• Rising interest rates have affected borrowing costs, influencing real estate and corporate investments. • Index: S&P 500 | 2023 Start: 3,800 | 2023 End: 4,250 | % Change
	• Asset: Ethereum | 2023 Start Price: $1,200 | 2023 End Price: $1,900 | % Change: 58.3%	• -0.29		• 0.6583		
	• Market Trends: The S&P 500 and NASDAQ indices have shown mixed performance, with technology stocks leading gains while energy and traditional industries lagged.	• -0.306		• 0.8115		
	• Index: S&P 500 | 2023 Start: 3,800 | 2023 End: 4,250 | % Change: 11.8%	• -0.46		• 0.6279		
	• Table 1: S&P 500 and NASDAQ Performance. Rising interest rates have affected borrowing costs, influencing real estate and corporate investments.	• -0.58		• 0.578		

Figure 18: Example of full explainability analysis for a financial RAG query.

8.3 Examples of pipeline realization

Table 18 demonstrates a complete walkthrough of our explainable RAG pipeline applied to a financial document given to it. Each component in the table corresponds to a distinct stage of our pipeline—retrieval, generation, hallucination detection, ablation testing, and rationale extraction—highlighting the transparency and interpretability enabled at every step.

- **Context and Retrieval (Similarity Score):** The top-k chunks retrieved using the SVM-based method are listed, along with their similarity scores. Chunks with higher similarity (lower absolute distance) are closer to the decision boundary. The chose of SVM was taken by default to satisfy diversity and similarity at the same time(in the case of small documents (our case) we will approximatively have same retrieved chunks based on cosine and SVM methods)

- **Generated Answer:** The response generated by DeepSeek R1 successfully integrates relevant financial signals, referencing borrowing costs, real estate impact, and index performance. However, it includes the claim that *"NASDAQ gained 72.1% in 2023"*, which appears suspicious and potentially hallucinated.

- **Ablation Testing:** By removing each context individually and recomputing semantic similarity with the full response, we estimate the influence of each chunk. Notably, with chunk 5 we got the highest drop in the score (0.578), which proves that this chunk is the most important in generating the response. Thus, deleting this chunk will alter significantly the semantic meaning of the generated response. This confirming the method's pivotal role in shaping the answer.

- **Hallucination Detection:** The hallucination detection module correctly flags the phrase `"NASDAQ gained 72.1% in 2023"` as fabricated, which is critical in avoiding the propagation of misinformation in financial contexts.

- **Rationale Extraction:** The LLM-based rationale extractor isolates the exact context segments that support the most the grounded part of the response. It correctly highlights the real estate impact chunk and the S&P 500 statistics, thus offering a transparent justification for the generated answer.

This example validates the effectiveness of each module in the pipeline:

- Retrieval supports both relevance and semantic diversity via SVM scoring.

- Generation leverages the retrieved context while still being subject to verification.
- Hallucination detection and rationale extraction enhance safety and interpretability.
- Ablation testing quantifies individual context contributions, promoting traceability.

Overall, this tightly integrated architecture empowers users to trust, audit, and explain each answer in high-stakes financial environments.

References

[1] Swapnaja Achintalwar, Adriana Alvarado Garcia, Ateret Anaby-Tavor, Ioana Baldini, Sara E. Berger, Bishwaranjan Bhattacharjee, Djallel Bouneffouf, Subhajit Chaudhury, Pin-Yu Chen, Lamogha Chiazor, Elizabeth M. Daly, Kirushikesh DB, Rogério Abreu de Paula, Pierre Dognin, Eitan Farchi, Soumya Ghosh, Michael Hind, Raya Horesh, George Kour, Ja Young Lee, Nishtha Madaan, Sameep Mehta, Erik Miehling, Keerthiram Murugesan, Manish Nagireddy, Inkit Padhi, David Piorkowski, Ambrish Rawat, Orna Raz, Prasanna Sattigeri, Hendrik Strobelt, Sarathkrishna Swaminathan, Christoph Tillmann, Aashka Trivedi, Kush R. Varshney, Dennis Wei, Shalisha Witherspoooon, and Marcel Zalmanovici. Detectors for Safe and Reliable LLMs: Implementations, Uses, and Limitations, August 2024. arXiv:2403.06009 [cs].

[2] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

[3] Xingyu Bai, Taiqiang Wu, Han Guo, Zhe Zhao, Xuefeng Yang, Jiayi Li, Weijie Liu, Qi Ju, Weigang Guo, and Yujiu Yang. Recouple Event Field via Probabilistic Bias for Event Extraction, May 2023. arXiv:2305.11498 [cs].

[4] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I. Jordan. Distribution-Free, Risk-Controlling Prediction Sets, August 2021. arXiv:2101.02703 [cs].

[5] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. RAGAS: Automated Evaluation of Retrieval Augmented Generation, September 2023. arXiv:2309.15217 [cs].

[6] Robert Friel, Masha Bebyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2025.

[7] Galileo Research. RAGBench Dataset on HuggingFace. https://huggingface.co/datasets/rungalileo/ragbench, 2025.

[8] Roni Goldshmidt and Miriam Horovicz. TokenSHAP: Interpreting Large Language Models with Monte Carlo Shapley Value Estimation, July 2024. arXiv:2407.10114 [cs].

[9] Christina Dan Wang Hongyang Yang, Xiao-Yang Liu. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023.

[10] Gautier Izacard, Myle Ott Hosseini, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

[11] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, April 2021. arXiv:2005.11401 [cs].

[12] Dina Mardaoui and Damien Garreau. An Analysis of LIME for Text Data, July 2021. arXiv:2010.12487 [stat].

[13] Sacha Muller, António Loison, Bilel Omrani, and Gautier Viaud. GroUSE: A Benchmark to Evaluate Evaluators in Grounded Question Answering, January 2025. arXiv:2409.06595 [cs].

[14] Cheng Niu, Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Randy Zhong, Juntong Song, and Tong Zhang. RAGTruth: A Hallucination Corpus for Developing Trustworthy Retrieval-Augmented Language Models, May 2024. arXiv:2401.00396 [cs].

[15] OpenAI. GPT-4 Technical Report, March 2024. arXiv:2303.08774 [cs].

[16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”Why Should I Trust You?”: Explaining the Predictions of Any Classifier, August 2016. arXiv:1602.04938 [cs].

[17] Junnan Wang, Zhe Gan, Jing Liu, Zicheng Liu, Lijuan Wang, Yejin Wang, and Yu Cheng. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[18] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, January 2023. arXiv:2201.11903 [cs].