



Self-Auditing Context Engine for AI Systems

Team - Hitansh_rg : Saatvik Sachdeva, Hitansh Gandhi, Dhyaan Shah

Basic Working Model for the Project : [GitHub Repository](#)

Abstract

Modern RAG systems generate prompts by selecting and discarding contextual information/data in the background due to which the users remain unaware of what data influenced the final output. This may pose risk in regulated environments as the system may give incorrect/incomplete answers leading to flawed decisions.



This project proposes a Self-Auditing Context Engine for AI systems that explicitly outputs and explains the context selection process. The system retrieves multiple documents, creates data chunks, assigns relevance scores and gives clear explanation for why some specific data chunks were used to generate the answer. Along with the final AI-generated answer, the model also outputs a human-readable context audit log that details selection decisions, rejection reasons, and relevance indicators. By exposing how context is selected and rejected, the system removes the “black box” problem, making AI responses more trustworthy, easier to debug, and more reliable than standard RAG approaches.

Introduction

Large Language Models (LLMs) are being widely used across various AI applications and have demonstrated strong capabilities in understanding and generating natural language. However, when an LLM is used independently, without any external support or grounding mechanism, it faces several limitations such as reliance on static training data, lack of domain-specific knowledge, and the generation of confident but unverifiable responses. These issues make standalone LLMs unreliable for real-world and enterprise use cases where correctness and accountability are important.

Retrieval-Augmented Generation (RAG) is an effective approach to overcome these limitations by generating responses based on externally retrieved documents. By incorporating relevant contextual information during answer generation, RAG systems improve factual accuracy and reduce false answers. However, even though RAG-based systems perform better than these standalone LLMs, most of them operate as black boxes from the user’s perspective as the process of context retrieval and selection remains hidden, and users are only presented with the final answer, without any explanation of what information was used or ignored.

This lack of transparency becomes a significant problem in enterprise and regulated environments such as finance, healthcare, and compliance reporting. Incorrect or incomplete context selection can lead to inaccurate or misleading answers, and the absence of an audit trail makes it difficult to verify, debug, or trust the system’s output. In such scenarios, users require correct answers backed up with clear explanations justifying the answer.



This project addresses this problem by proposing a Self-Auditing Context Engine for AI systems. The system retrieves multiple context sources, assigns relevance scores, and explicitly records the reasons for including or excluding specific documents or chunks during answer generation. Along with producing the final answer, the system also generates a human-readable context audit log that explains the selection decisions in a clear and understandable manner. The goal of this project is to demonstrate that making context selection transparent and explainable can significantly improve the trustworthiness, debuggability, and reliability of RAG-based AI systems, particularly in enterprise and high-stakes environments.

Problem Statement

The objective of this project is to design and implement a self-auditing AI system that not only generates answers to user queries but also explains how the contextual information used for those answers was selected. The system retrieves multiple documents or data chunks, evaluates their relevance, and records explicit reasons for including or excluding each context source during answer generation.

The system is designed to produce two outputs: a final AI-generated response and a human-readable context audit log. The audit log clearly describes which documents were considered, why certain information was selected, and why other data was ignored. By making the context selection process transparent and traceable, the system aims to improve trust, accountability, and reliability of AI-generated answers, particularly in enterprise and regulated environments where explainability is essential.

System Overview

The system follows a multi-stage workflow process :

Step1 : Document Access

The user provides the system with access to a collection of documents or records. These documents act as the complete knowledge source for the system and are used for context retrieval during answer generation.

Step2 : User Query

The user submits a natural-language query. The system is required to answer the query based only on the available documents and must not rely on external or assumed knowledge.

Step3 : Context Retrieval and Scoring

The user query is processed using a retrieval or similarity-based mechanism. Each document or chunk is evaluated against the query and assigned a relevance score. Based on these scores, the system identifies a set of candidate context sources that may contribute to the final answer.

Step4 : Context Selection and Decision Logging

From the retrieved candidates, the system applies predefined selection rules or thresholds to determine which documents or chunks should be included in the answer-generation process. At the same time, the system logs explicit reasons for each decision, recording why certain context sources were included and why others were excluded.

Step5 : Answer Generation

The selected context is passed to a language model for answer generation. The model generates a response strictly grounded in the selected context, ensuring that the output is supported by the retrieved information.

Step6 : Context Audit Log Generation

Alongside the final answer, the system generates a human-readable context audit log. This log explains which documents were considered, which were selected, which were rejected, and the reasoning behind each decision, including relevance indicators where applicable.

Step7 : Output

The system outputs two components: a final natural-language answer to the user's query and a corresponding context audit log that provides transparency into the context selection process and supports verification and trust in the generated response.

Methodology

Context Retrieval Module

Document Loading

Our model asks the user for the file path to access the folder which contains the PDFs - FileSystemBlobLoader is used to identify the PDF files. Now, PyPDFParser is used to extract the raw text from these PDFs for further processing. While loading files in the Python script, we create a list of documents each storing one page of files.

Splitting into Chunks

Now, we need to split these documents into chunks because we are using the BAAI/bge-small-en embedding model which is based on BERT. BERT models have a hard limit of 512 tokens (approximately 300-400 words) so if we do not split our documents into chunks the embedding model would read the first 300-400 words and truncate off the rest.

For this model, we are using RecursiveCharacterTextSplitter instead of normal text splitter because using a normal text splitter would cut off the lines in between, but the recursive text splitter ensures not to cut the lines off in between - splitting first by paragraphs, then by sentences, and finally by words. Each chunk of ours contains 500 characters of which 50 characters are overlapping to give context to the script so that they do not get scrambled.

Context Scoring and Retrieval

Relevance Scoring

Once the chunks are created, each chunk is converted into a vector representation using an embedding or similarity-based mechanism. When a user query is received, it is also converted into a vector representation. The system then computes similarity scores between the query vector and each document chunk vector.



These scores act as relevance indicators and are used to determine how closely each chunk aligns with the user's query. All chunks are retained at this stage along with their associated scores to support later auditing and explanation.

Candidate Context Selection

Based on the relevance scores, the system selects a subset of top-ranking chunks as candidate context sources. This selection is not final and serves only as an intermediate step before explicit inclusion or exclusion decisions are made.

Context Selection and Decision Logging

Selection Rules and Thresholds

The system applies predefined rules or thresholds to the candidate chunks in order to decide which context sources should be used for answer generation. These rules may include relevance score cutoffs, document freshness, or other configurable constraints.

Decision Logging

A key component of the methodology is decision logging. For each document or chunk, the system records:

- Whether it was included or excluded
- The relevance score assigned to it
- The rule or condition that influenced the decision

This information is stored in a structured format to ensure traceability and explainability of the context selection process.

Answer Generation Module

Prepare Retrieved Context

Now, each retrieved chunk is assigned a unique ID. Chunk IDs are necessary for traceability.

Prompt Construction

A constrained prompt is passed to the LLM. The prompt includes the following things :

- User query
- Retrieved text chunks, each of which is labeled with its ID
- Explicit instruction enforcing : use of retrieved text only, prohibition to use external knowledge and citation of chunk ID's.

It is also specifically mentioned to LLM (in the prompt) that the output should strictly be in a JSON output format.

Answer Generation using LLM

The answer will be generated by a local LLM (here, LLaMA-3-8b via Ollama). The temperature is set to zero for deterministic output. (Temperature of LLM is a setting that controls randomness of text generation - so here temperature is set low to generate a focused and deterministic output).

Output Parsing and Validation

Now, the raw output from the language model is parsed as JSON and validated before acceptance. The validation check includes : checking if the output is a valid JSON or not and if the answer is present or not.

Context Audit Log Generation

After the answer is generated, the system compiles a context audit log. This log provides a human-readable explanation of:

- Which documents or chunks were considered
- Which were selected for answer generation
- Which were excluded and why
- Relevance scores or confidence indicators

The audit log is designed to be understandable by non-technical users while still accurately reflecting the system's internal decisions.

Final Output and Explainability

We show the user the LLM-generated answer along with a detailed context audit log to ensure transparency in the answer-generation process.

We provide the user a full text answer generated by the LLM model LLaMA-3-8b (Ollama). This answer is produced strictly using the selected context from the provided documents and does not rely on any external or assumed knowledge.

Alongside the answer, the system generates a Context Audit Log in a human-readable format. The audit log explains how the system evaluated the available documents and how context selection decisions were made. The log follows a format which includes:

- Documents or chunks considered by the system
- Relevance scores assigned to each source
- Context sources that were selected for answer generation
- Context sources that were excluded, along with explicit reasons

This allows the user to trace the origin of the answer and understand the reasoning behind the system's decisions.

Sample Output :

```
(venv) hitansh_2205@Hitanshs-MacBook-Air PS1_Self-Auditing Context Engine % python main.py
[...]
/Users/hitansh_2205/Documents/College/Projects/DataForge'26/PS1_Self-Auditing Context Engine/retriever.py:20: LangChainDeprecationWarning: Th
e class `HuggingFaceBgeEmbeddings` was deprecated in LangChain 0.2.2 and will be removed in 1.0. An updated version of the class exists in th
e `langchain-huggingface package and should be used instead. To use it run `pip install -U `langchain-huggingface` and import as `from `langc
hain_huggingface import HuggingFaceEmbeddings``.
    hf = HuggingFaceBgeEmbeddings(
Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster downloads.
Loading weights: 100%|██████████| 199/199 [00:00<00:00, 6076.71it/s, Materializing param=pooler.dense.weight]
BertModel LOAD REPORT from: BAAI/bge-small-en
Key           | Status   | |
---+---+---+
embeddings.position_ids | UNEXPECTED | |
Notes:
- UNEXPECTED : can be ignored when loading from different task/architecture; not ok if you expect identical arch.
Database found! Do you want to update it? (y/n): y
What is the folder path: /Users/hitansh_2205/Documents/College/Projects/DataForge'26/Final Project/Sample_Data
Ignoring wrong pointing object 27 0 (offset 0)
Ignoring wrong pointing object 28 0 (offset 0)
Creating vector database...
Database saved.
What is question you want to ask: What is Climate and Climate Change?
/Users/hitansh_2205/Documents/College/Projects/DataForge'26/Final Project/Sample_Data/Document_2_Climate_Change.pdf is highly relevant.
/Users/hitansh_2205/Documents/College/Projects/DataForge'26/Final Project/Sample_Data/Climate Change.pdf is relevant.

Final Answer :
Climate refers to the long-term weather patterns and conditions that exist in a particular region. Climate change, on the other hand, is the warming of the planet due to human activities that release greenhouse gases, such as carbon dioxide and methane, into the atmosphere. This poses serious risks to ecosystems and biodiversity, with many species struggling to adapt to rapidly changing environmental conditions, leading to habitat loss and extinction. Coral reefs, for example, are highly sensitive to temperature changes and are experiencing widespread bleaching due to warming oceans. Human societies are deeply affected by climate change, with food security threatened by changing weather patterns and rising temperatures.
```

(Sample user query : "What is Climate and Climate Change?")

Relevant Documents :

/Users/Documents/Sample_Data/Document_2_Climate_Change.pdf is highly relevant.

/Users/Documents/Sample_Data/Climate Change.pdf is relevant.

LLM-Generated Answer (Context Constrained)

Climate refers to the long-term weather patterns and conditions that exist in a particular region. Climate change, on the other hand, is the warming of the planet due to human activities that release greenhouse gases, such as carbon dioxide and methane, into the atmosphere. This poses serious risks to ecosystems and biodiversity, with many species struggling to adapt to rapidly changing environmental conditions, leading to habitat loss and extinction. Coral reefs, for example, are highly sensitive to temperature changes and are experiencing widespread bleaching due to warming oceans. Human societies are deeply affected by climate change, with food security threatened by changing weather patterns and rising temperatures.

The context audit log explicitly points to the document sources that contributed to the final answer. This enables users to verify the correctness of the response and understand how individual pieces of information influenced the output.

If relevant context is missing or incorrectly selected, the audit log makes this visible, allowing users to identify potential issues in retrieval or selection. In cases where the audit log shows weak or insufficient context but an answer is still produced, the user can easily detect possible hallucination or overgeneralization by the language model.

By exposing both the final answer and the reasoning behind context selection, the system provides a transparent and trustworthy framework for AI-assisted question answering, especially in enterprise and high-stakes environments.

Conclusion

In this project, we successfully designed and implemented a **Self-Auditing Context Engine** that addresses the lack of transparency in conventional Retrieval-Augmented Generation (RAG) systems. The primary objective of the



system was to move beyond generating answers alone and instead provide clear explanations of how contextual information was selected during the answer-generation process.

The proposed system ensures that every answer is supported by explicitly selected document context and is accompanied by a human-readable audit log. By exposing which documents were considered, which were selected, and why certain information was ignored, the system improves trust, accountability, and debuggability of AI-generated outputs. This approach is particularly valuable in enterprise and regulated environments, where understanding the reasoning behind AI decisions is as important as the correctness of the answer itself.

Overall, the project demonstrates that adding a self-auditing and explainability layer to RAG pipelines can significantly reduce the “black box” behavior of AI systems and make them more suitable for real-world decision-support applications.

Limitations

Heuristic-Based Context Selection

The current system relies on predefined rules and relevance thresholds for context selection. While this approach is effective and interpretable, it may not always capture nuanced relationships between documents and queries, especially in complex or ambiguous scenarios.

Dependence on Similarity Scoring

Context relevance is primarily determined using similarity-based scoring mechanisms. In some cases, semantically important information may receive lower similarity scores and be excluded, even though it could contribute to a more complete answer.

Limited Reasoning Beyond Retrieved Context

The system intentionally restricts the language model to use only selected context to avoid hallucinations. While this improves reliability, it may also limit the system’s ability to provide richer explanations when the available documents are sparse or incomplete.

Scalability of Audit Logs

As the number of documents increases, the context audit log may become lengthy and harder for non-technical users to interpret. Additional summarization or visualization techniques may be required to maintain clarity at scale.

References

- Langchain documentation : <https://docs.langchain.com/>
- spaCy documentation : <https://spacy.io/usage/spacy-101#features>
- Python JSON library documentation :
<https://docs.python.org/3/library/json.html>
- Python requests library documentation :
<https://docs.python.org/3/library/json.html>