

White Paper

A Memory-Aware AI Assistant for Knowledge Retrieval

A context-aware RAG system that enhances knowledge retrieval through conversational memory and semantic search

Date: 27/07/2026

AI/ML

Ganeshan K

N-KORR Technologies Pvt Ltd.

1 Executive Summary

Large Language Models (LLMs) have changed the way people interact with digital knowledge by enabling natural language conversations and intelligent question answering. However, many Retrieval-Augmented Generation (RAG) systems focus only on the user's latest question, without considering previous conversations or the user's intent. As a result, follow-up questions can be misunderstood, leading to less relevant retrieval and inconsistent responses.

We present a Memory-Aware AI Assistant, an architecture that enhances RAG by incorporating conversational memory, query understanding, and context-aware retrieval. Instead of relying only on the latest user message, we leverage conversation history and indexed document knowledge to better understand user intent and retrieve the most relevant information.

Our architecture integrates document ingestion, semantic search, query rewriting, reranking, and response generation into a modular workflow. By combining document knowledge with conversational memory, the assistant delivers more accurate, context-aware, and reliable responses, making it well-suited for knowledge retrieval, technical support, research assistance, and policy guidance.

2 Problem in Context

Many RAG systems process each user query independently, using only the latest message to retrieve relevant information from a vector database. While this works well for standalone questions, it becomes less effective during multi-turn conversations where users ask follow-up questions or refer to previous topics without repeating the full context.^{[1][2]}

As a result, the system may retrieve irrelevant information, misunderstand the user's intent, or generate incomplete responses. These issues reduce the overall user experience and limit the assistant's ability to support natural, continuous conversations.

To address these challenges, retrieval should consider both the current query and the conversation history. By incorporating conversational memory, query rewriting, and context-aware retrieval, the system can better understand user intent and retrieve more relevant information.

Our architecture integrates session memory with retrieval and response generation, enabling the assistant to deliver accurate, context-aware, and grounded responses across multi-turn conversations.

Example Without Memory Awareness

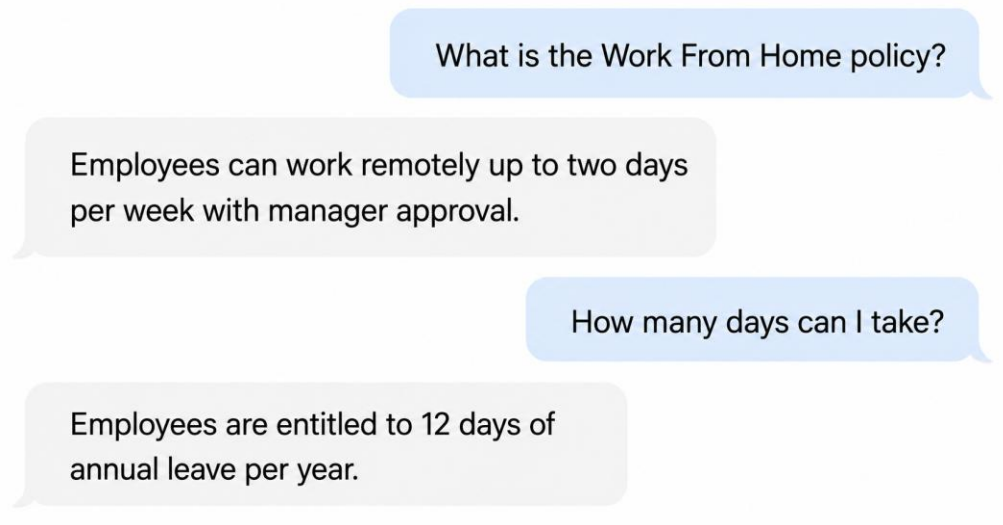


Figure 2.1 Example Without Memory Awareness

Because the system processes the second query independently, it interprets "days" as a request about leave policies rather than the previously discussed Work From Home policy. As a result, it retrieves information about annual leave instead of Work From Home, producing an incorrect response.

Example With Memory Awareness

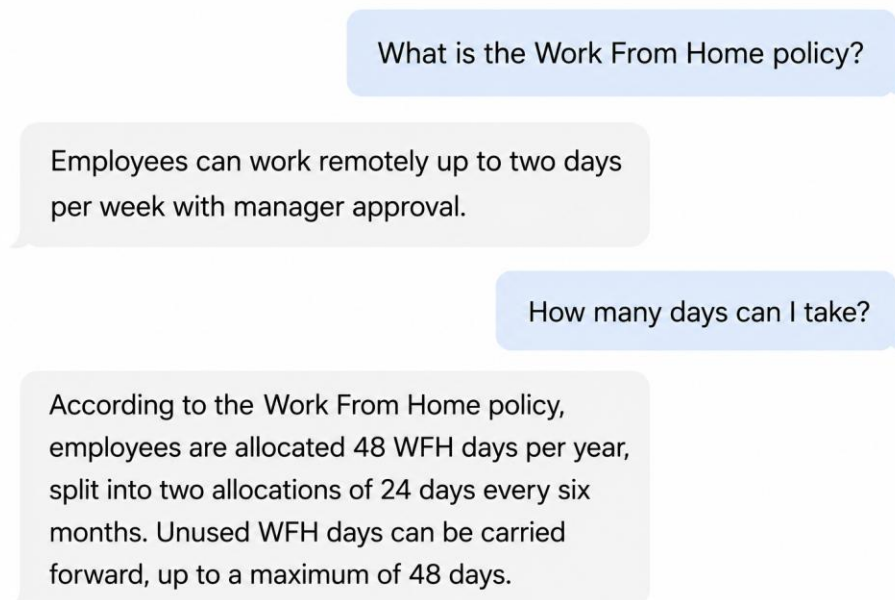


Figure 2.2 Example With Memory Awareness

The assistant uses the conversation history to understand that the user is referring to the Work From Home policy. It rewrites the query as,

How many Work From Home days are employees allowed

The rewritten query retrieves the correct document, allowing the assistant to provide an accurate and context-aware response.

3 Technical Deep Dive

3.1 Architectural Overview

The proposed Memory-Aware AI Assistant extends a conventional Retrieval-Augmented Generation (RAG) architecture by integrating conversational memory into the knowledge retrieval process. Instead of processing each user request independently, the system combines conversation history, intelligent query understanding, semantic retrieval, and grounded response generation to deliver context-aware answers.

The architecture follows a modular design in which each component is responsible for a specific stage of the pipeline. Documents are first transformed into searchable vector representations during ingestion. During user interaction, the assistant analyzes the incoming query, determines whether external knowledge is required, retrieves the most relevant document chunks, and generates a response using both retrieved knowledge and conversation memory. Finally, the interaction is persisted to support future conversations.

This modular approach improves maintainability, scalability, and retrieval accuracy while reducing hallucinations through evidence-based response generation.

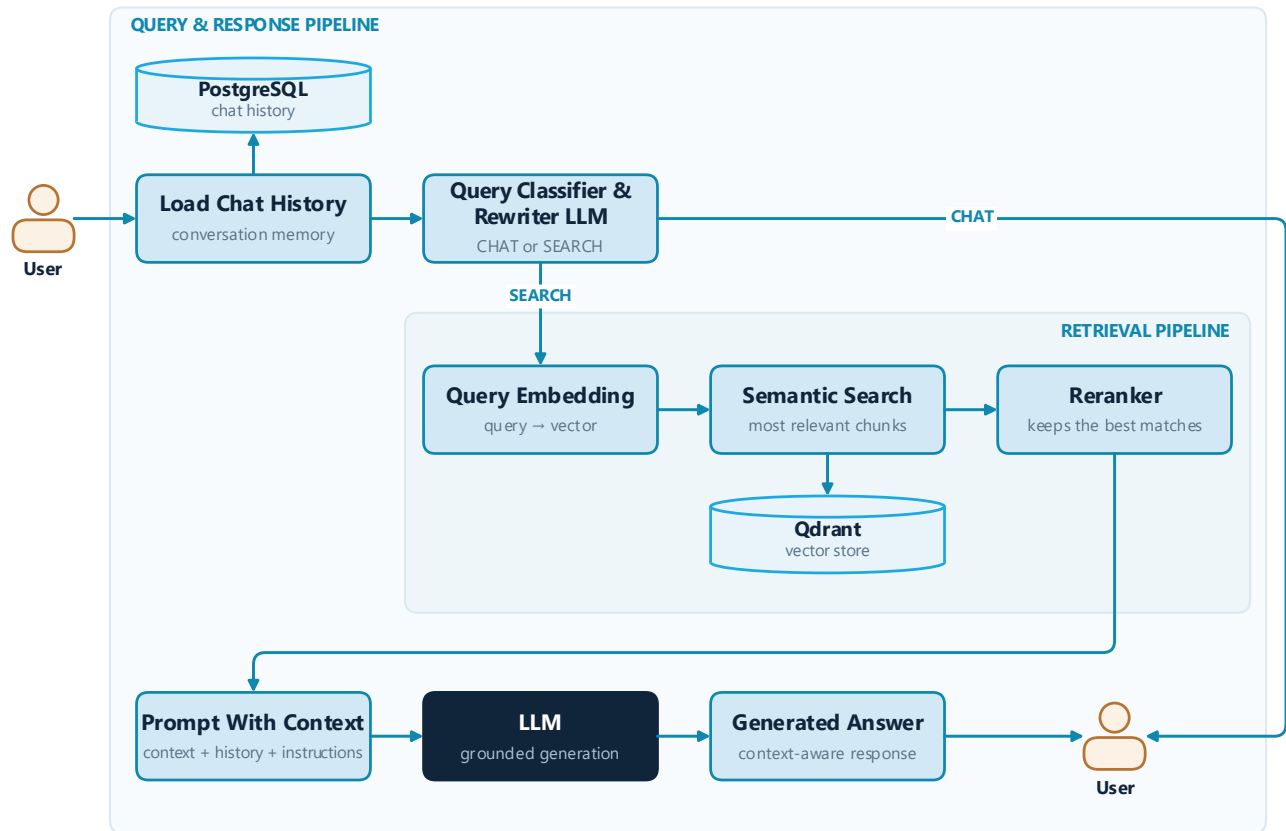


Figure 3.1. The RAG pipeline of query processing and answer generation

3.2 Document Ingestion Pipeline

The ingestion pipeline converts raw documents into structured knowledge that can be efficiently searched using semantic similarity. Instead of indexing entire documents, content is divided into smaller overlapping chunks that preserve contextual continuity while improving retrieval precision.

Each chunk is transformed into a numerical embedding and stored together with its metadata in the vector database. This preprocessing stage is performed only once during document upload, allowing subsequent user queries to be answered efficiently.

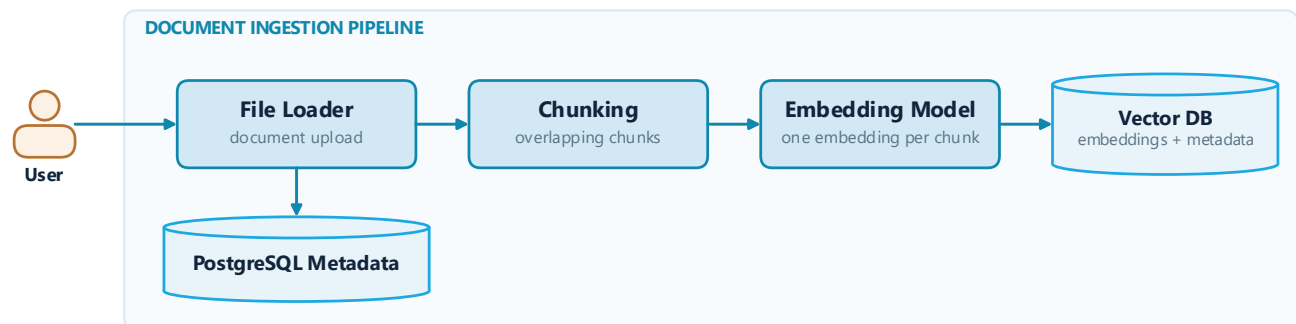


Figure 3.2. The RAG Ingestion pipeline

Each document is divided into overlapping chunks before embeddings are generated, improving retrieval accuracy while preserving contextual continuity.

Each chunk is represented as:

```
{  
  "chunk_id": "...",  
  "chunk_index": "...",  
  "document_id": "...",  
  "document_type": "...",  
  "page": "...",  
  "source": "...",  
  "content": "..."  
}
```

3.3 Memory-Aware Retrieval

Most RAG systems retrieve information using only the user's latest query, which can make it difficult to understand follow-up questions or references to earlier parts of the conversation.

The proposed architecture improves retrieval by incorporating conversation history into the search process. Before retrieving documents, the system uses the current query together with previous interactions to better understand the user's intent.

The contextualized query is then converted into an embedding and used to search the vector database for relevant document chunks. The retrieved results are reranked to select the most relevant information before being passed to the language model.

By combining conversational memory with semantic retrieval, the assistant delivers more accurate, context-aware responses and maintains continuity throughout multi-turn conversations.



Figure 3.3: Context-Aware Retrieval Workflow

3.4 Query Understanding

Before retrieval begins, the system analyzes the user's latest message to determine whether external knowledge retrieval is required. Not every user interaction needs document search, many requests are simple conversational exchanges such as greetings, acknowledgements, or follow-up discussions.

To optimize performance, the Query Understanding module classifies every incoming request into one of two categories:

CHAT – General conversational messages that can be answered directly by the language model.

SEARCH – Questions that require information retrieved from the vector database.

For **SEARCH** requests, the Query Rewriter reconstructs incomplete or context-dependent questions into standalone search queries by utilizing the conversation history. This ensures that semantic retrieval is performed using a complete and meaningful query, resulting in more accurate document retrieval.[3]

By separating conversational requests from knowledge retrieval, the system reduces unnecessary searches, improves response latency, and ensures that retrieval resources are used only when required.

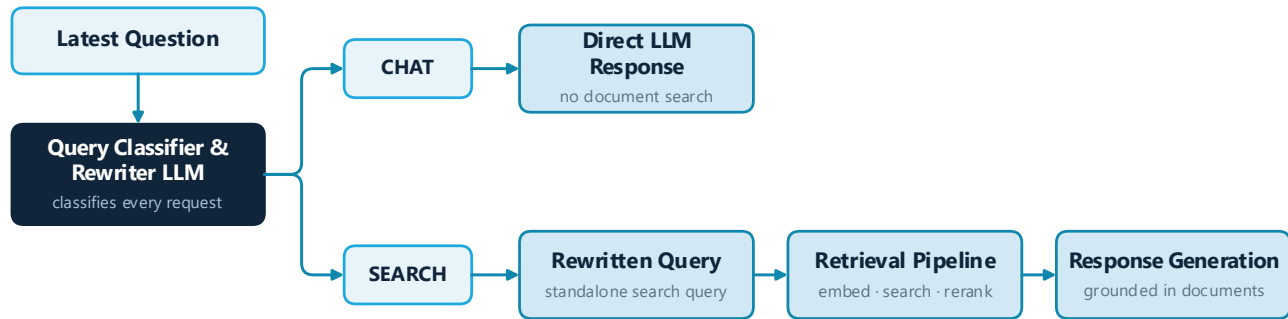


Figure 3.4: Query Understanding Workflow

3.5 Retrieval and Response Generation

Once the query has been rewritten, it is converted into an embedding and used to search the vector database for the most relevant document chunks. The retrieved results are then reranked to select the most relevant information.

The selected document context, together with the conversation history and system instructions, is provided to the Large Language Model (LLM) to generate the final response.

By combining retrieved knowledge with conversational context, the assistant produces responses that are accurate, relevant, and grounded in the available documents, while reducing the likelihood of hallucinations.

4 NKORR's Approach

At NKORR, we see conversational context as a fundamental requirement for business AI assistants. In most business environments, users rarely ask standalone questions. They ask follow-up questions, refer to earlier discussions, and expect the assistant to understand the conversation without requiring the same context to be repeated. This is one of the key limitations of conventional RAG systems, where retrieval is often driven only by the latest user query.

To address this, our architecture combines session memory with query rewriting before retrieval. Rather than sending the latest message directly to the vector database, the assistant first considers the conversation history to identify the user's actual intent. The resulting contextualized query is then used to retrieve and rerank the most relevant document chunks before response generation. This allows retrieval to remain grounded in enterprise knowledge while preserving continuity across multi-turn conversations.

This design has practical value across a range of business scenarios. Customer support teams can resolve follow-up questions without asking users to restate previous issues, improving both response quality and support deflection. Employees can search internal documentation through natural conversations instead of repeatedly providing background information. The same approach also supports compliance and policy guidance, where accurate interpretation depends on both the current question and the context established earlier in the conversation.

Our objective is not simply to retrieve documents or generate responses, but to combine both capabilities in a way that reflects how people naturally interact with organizational knowledge. By incorporating conversational memory

into the retrieval process, the assistant delivers responses that remain relevant, grounded, and consistent throughout the conversation.

5 Recommendations

Organizations planning to implement or expand Retrieval-Augmented Generation (RAG) solutions should consider the following practices:

- **Evaluate multi-turn conversations, not just single queries:-** Test the assistant with follow-up questions and contextual references to identify retrieval failures before deploying at scale.
- **Treat conversational memory as a core architectural component:-** Incorporate session memory, query rewriting, and reranking into the retrieval pipeline rather than relying solely on prompt engineering to preserve context.
- **Assess solutions using real business scenarios:-** Evaluate AI assistants with representative use cases such as customer support, internal knowledge retrieval, and compliance guidance, where maintaining conversational context is critical to response quality.
- **Strengthen retrieval with complementary techniques:-** Combine semantic and keyword search through hybrid retrieval, and use conversation summarization to maintain context while reducing language model input size during long interactions.

By adopting these practices, organizations can improve retrieval accuracy, maintain conversational continuity, and deliver more reliable AI-assisted experiences across enterprise knowledge workflows.

6 References

#	Source	Notes
1	https://direct.mit.edu/tacl/article/doi/10.1162/TACL.a.19132114/mtRAG-A-Multi-Turn-Conversational-Benchmark-for	mtRAG: A Multi-Turn Conversational Benchmark for Evaluating Retrieval-Augmented Generation Systems
2	https://arxiv.org/abs/2312.10997	Retrieval-Augmented Generation for Large Language Models: A Survey
3	https://arxiv.org/html/2406.18960v1	A Surprisingly Simple yet Effective Multi-Query Rewriting Method for Conversational Passage Retrieval