GraphRAG with ontotext GraphDB and Neo4j

Kerfalla Cissé, Antonio Dantas

University of Geneva - Centre universitaire d'informatique

20 september 2024

Bio

Master in systems and services - Knowledge engineering

Bachelor project : Chatbot connected to rdf knowledge graph on restaurants. The agent broadens his knowledge and refines conversations with users. Sources : Geonames, DBpedia, OpenstreetMap.

Linkedin

Master in systems and services - Knowledge engineering

Bachelor project : Evaluating the Effectiveness of AI-driven Systems in Personalizable Information Extraction and Retrieval : A Study on Combining LLMs, Knowledge Graphs, and Word Embeddings for Industry-Wide Knowledge Management.

Linkedin

Table of contents

Introduction

GraphRAG

Implementation

Demo

References

What is RAG

Definition

"Retrieval-Augmented Generation (RAG) has emerged as a promising solution by incorporating knowledge from external databases. This enhances the accuracy and credibility of the models, particularly for knowledge-intensive tasks, and allows for continuous knowledge updates and integration of domain-specific information" [1].

RAG workflow

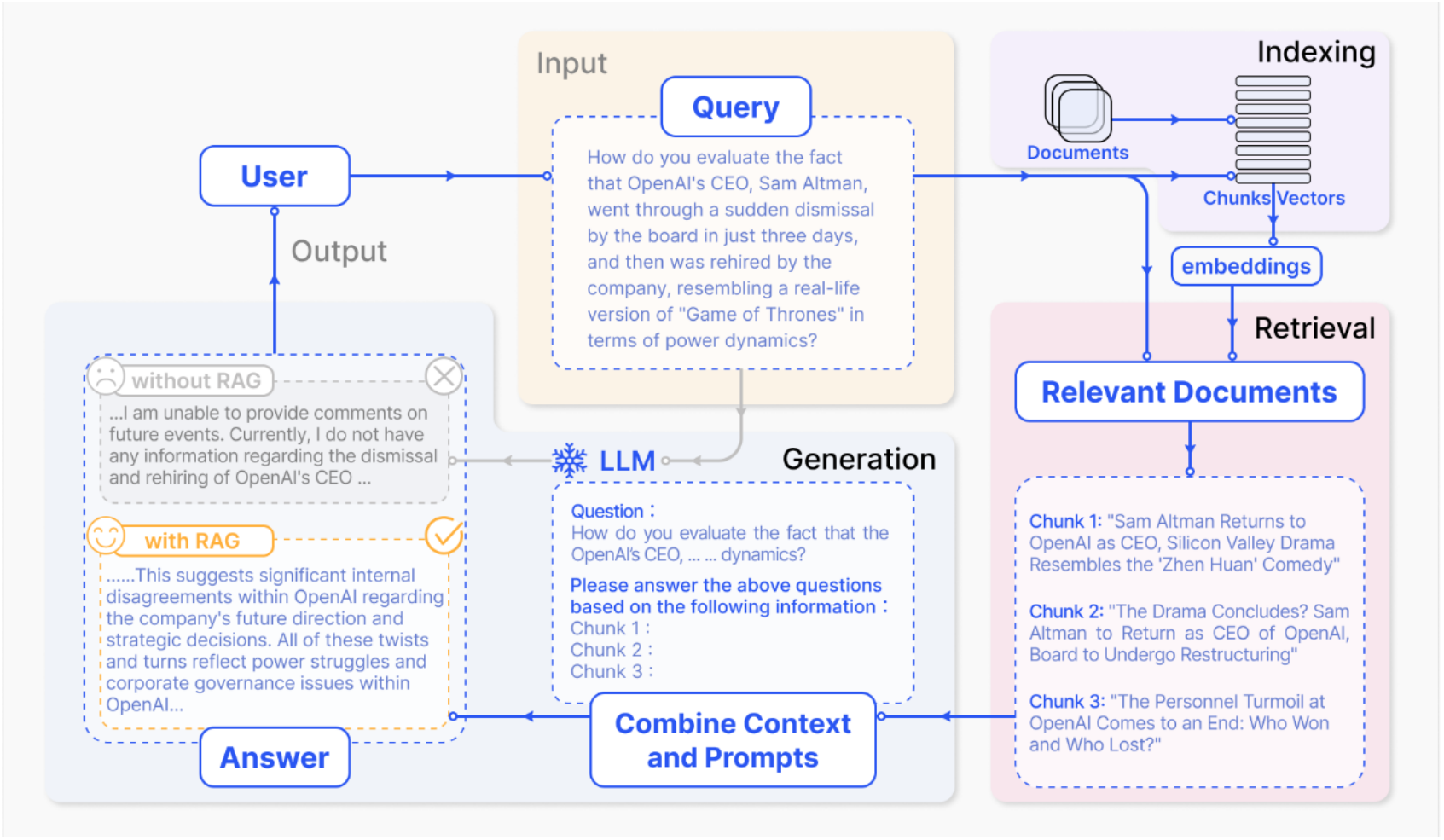


Figure – Source [1].

RAG key steps

1. The corpus is partioned into chunks, upon which vector indices are constructed utilizing an encoder model (offline)
2. RAG identifies and retrieves chunks, based on their vector similarity to the query and indexed chunks
3. The model synthesizes a response conditioned on the contextual information gleaned from the retrieved chunks

Advantage : RAG's key advantage lies in its obviation of the need for retraining of LLMs for task specific applications [1].

Types of RAG

While RAG were cost-effective and surpassed the performance of the native LLM, they also exhibited **several limitations**.

There are three types of RAG : Naive RAG, Advanced RAG and Modular RAG

Naive RAG

1. Represents the earliest methodology. It follows traditional process that include indexing, retrieval and generation [1]
2. **Drawback in Naive RAG** [1] :
   - **Retrieval** quality (low precision), can lead to potential hallucination
   - Low recall also occurs, resulting in the failure to retrieve all relevant chunks
   - Response **generation** quality presents hallucination challenge where the model generates answers not grounded in the provided context
   - The **augmentation** process presents its own challenges in effectively integrating context from retrieved passages with the current generation task

Advanced RAG

Advanced RAG has been developed to address shortcomings of Naive RAG [1].

- **Pre-retrieval** : integration of sophisticated dynamic embedding model that captures contextual understanding
  - OpenAI's embeddings-ada-02 model
  - BERT
  - ...
- **Post-retrieval** : re-raking the retrieved information to relocate the most relevant content to the edges of the prompt

Modular RAG

Provides greater versatility and flexibility. It's becoming the norm in the RAG domain [1].

- Enhance functional modules, such as incorporating a search module for similarity, etc.
- The core components of RAG framework is divided as several modules

Comparison between the three paradigms of RAG

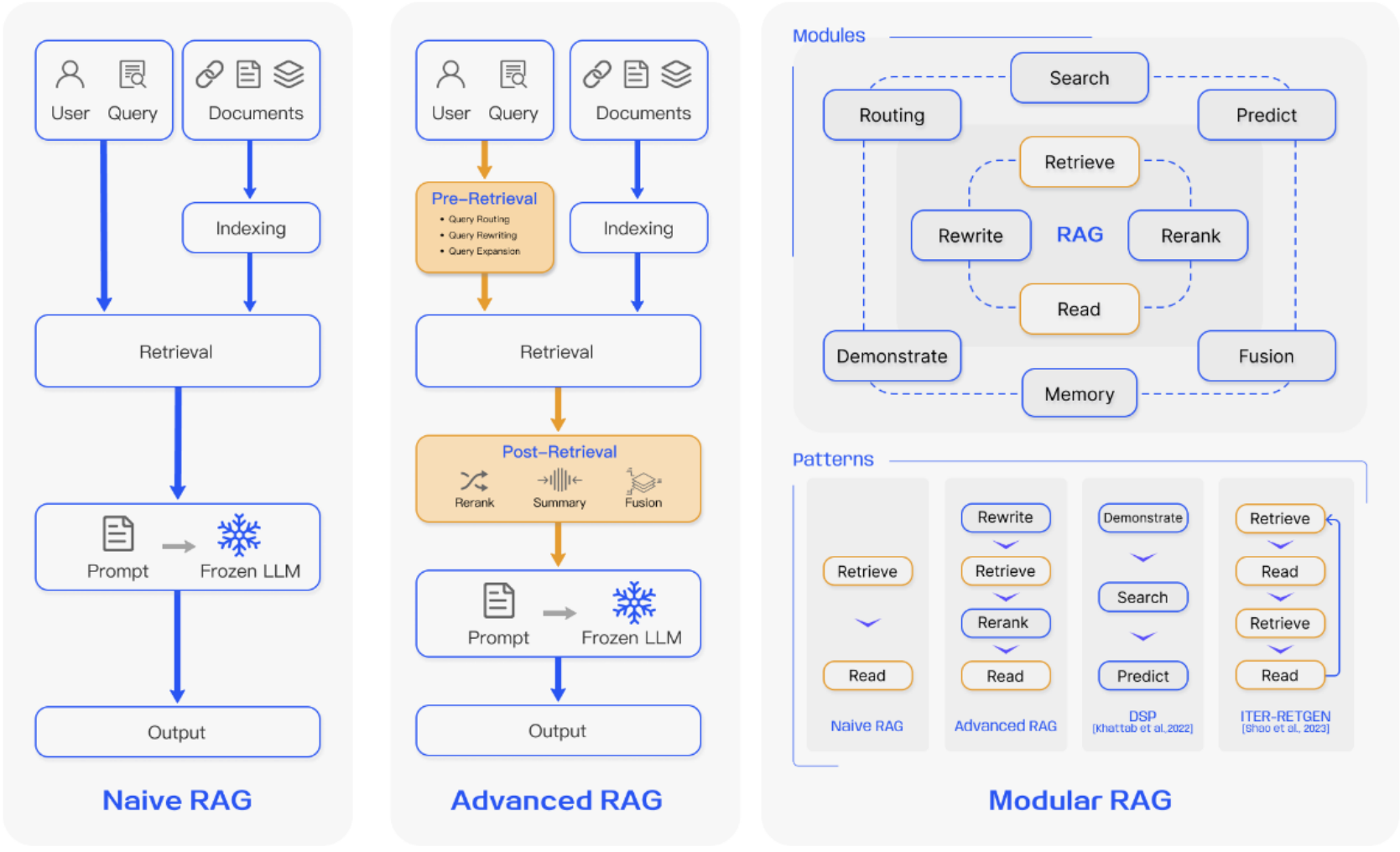


Figure – Source [1]

GraphRAG

Definition

Graph RAG is an enhancement over the popular RAG approach. Graph RAG includes a graph database as a source of the contextual information sent to the LLM. Providing the LLM with textual chunks extracted from larger sized documents can lack the necessary context, factual correctness [...] With Graph RAG each record in the vector database can have contextually rich representation increasing the understandability of specific terminology, so the LLM can make better sense of specific subject domains [2].

Types of GraphRAG [2]

- **Graph as a Content Store** : extract relevant chunks of documents and ask LLM to answer using them. It requires a KG containing relevant textual content and metadata as well as integration with a vector database
- **Graph as a subject Matter expert** : extract description of concepts and entities relevant to the natural language (NL) question and pass those to the LLM as additional "semantic context". It requires entity linking or another mechanism
- **Graph as Database** : map (part of) the NL question to a graph query, execute the query and ask the LLM to summarize the results. The graph should hold relevant factual information. It also requires entity linking and NL-to-Graph-tool

Indexing

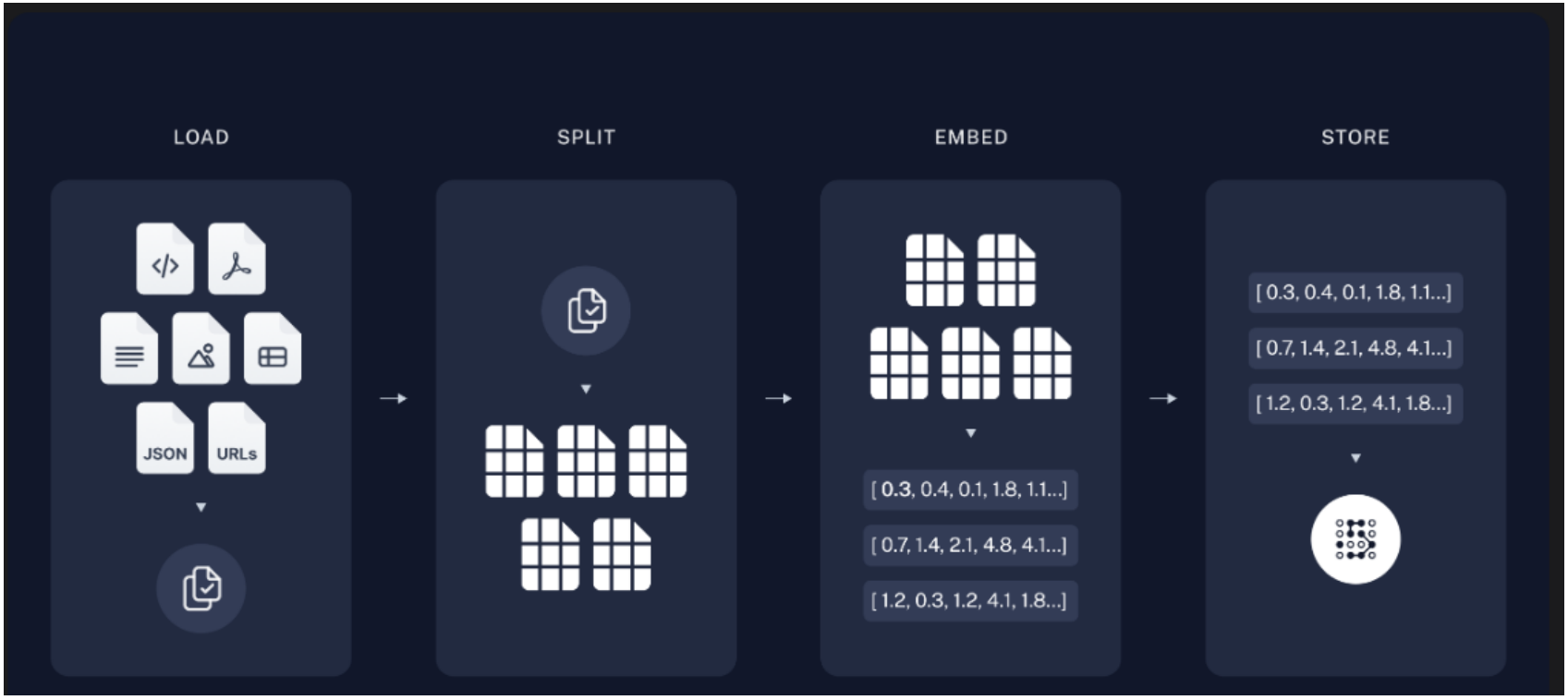


Figure – Source [3]

Retrieval and Generation

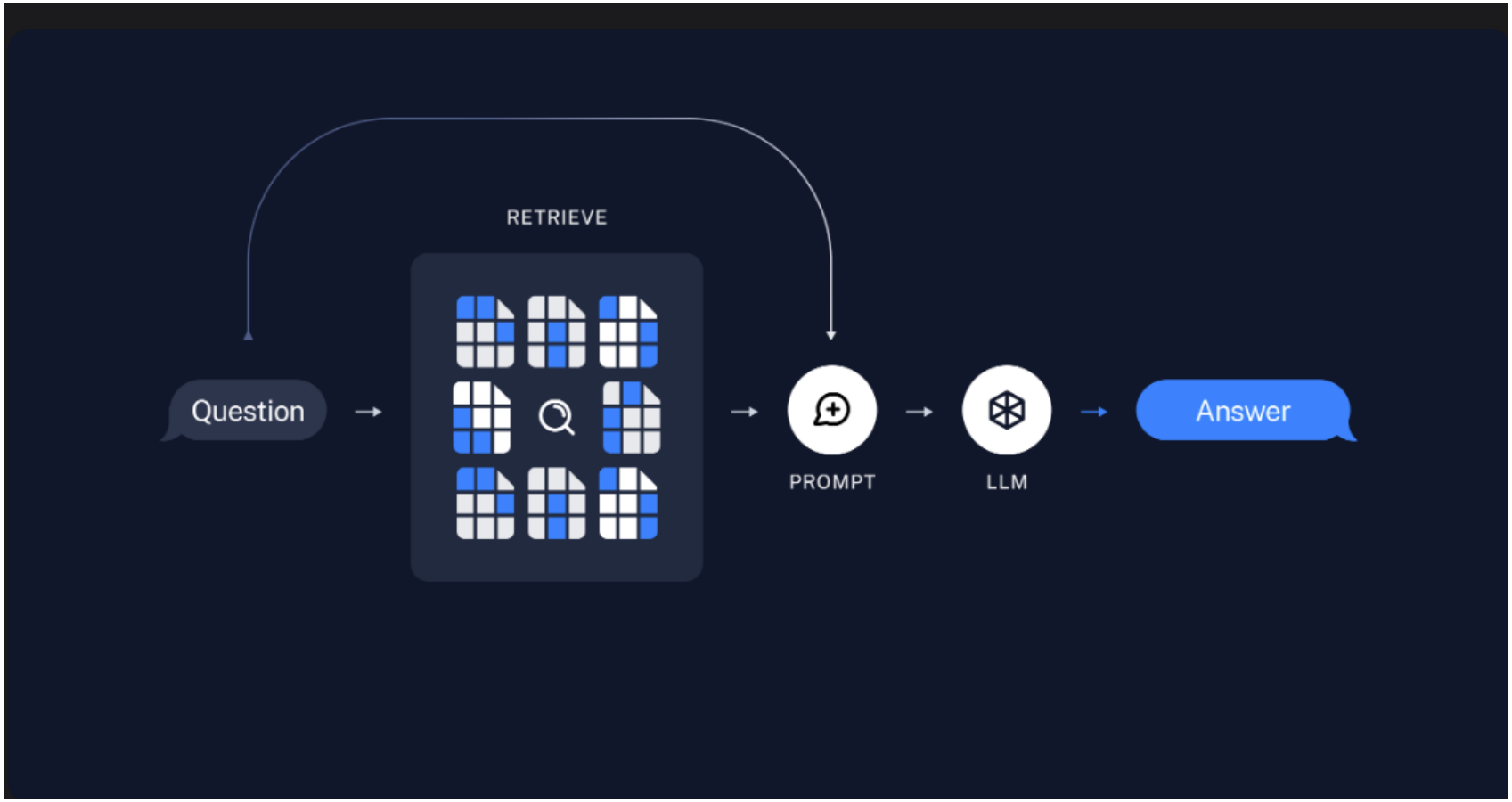
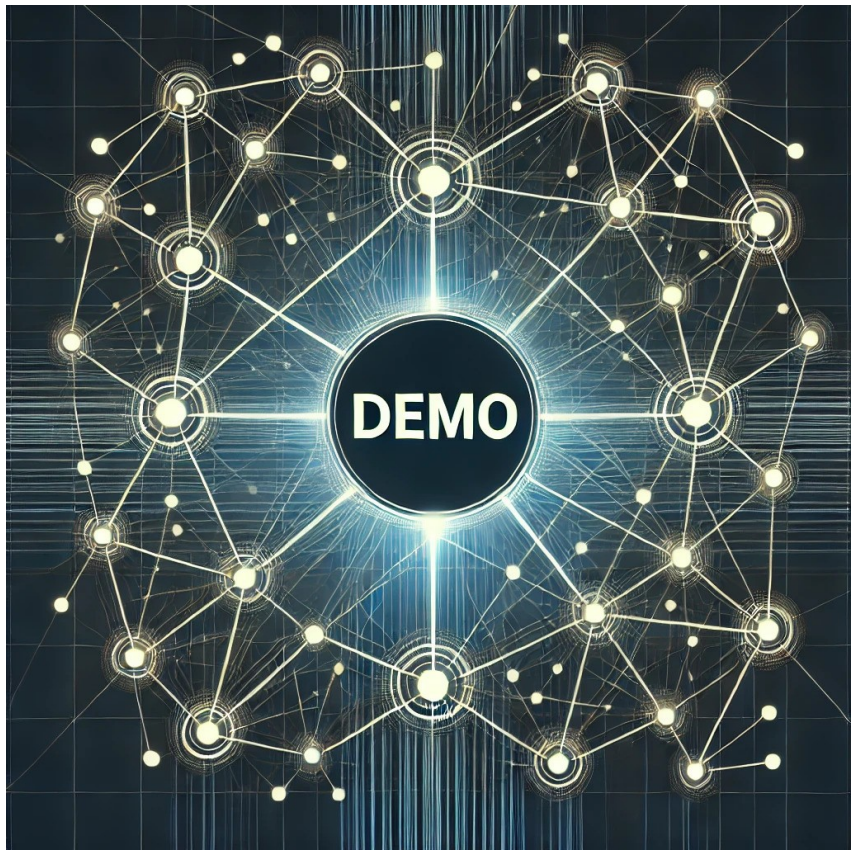


Figure – Source [3].

Demo



Bibliography

[1] Y. Gao, Y. Xiong, X. Gao et al., *Retrieval-augmented generation for large language models : a survey*, 4 jan. 2024. arXiv : 2312.10997[cs]. adresse : http://arxiv.org/abs/2312.10997 (visité le 07/03/2024).

[2] Ontotext-GraphDB, *What Is Graph RAG ?* Adresse : https://www.ontotext.com/knowledgehub/fundamentals/what-is-graph-rag/.

[3] langchain, *Build a Retrieval Augmented Generation (RAG) App*, adresse : https://python.langchain.com/docs/tutorials/rag/.