9 Advanced RAG Techniques

To build production ready apps

Joanna Stoffregen

co-founder & AI Product Lead
@Labsbit.ai

A **naive RAG system** lacks the sophistication needed for production-level performance

Let's look into some **advanced techniques** to make the system production ready

A production RAG system is split into **3 main components**

1

Ingestion

Load embeddings into the vector database

2

Retrieval

Query the vector database for relevant content

3

Generation

Use the retrieved context to answer user's query

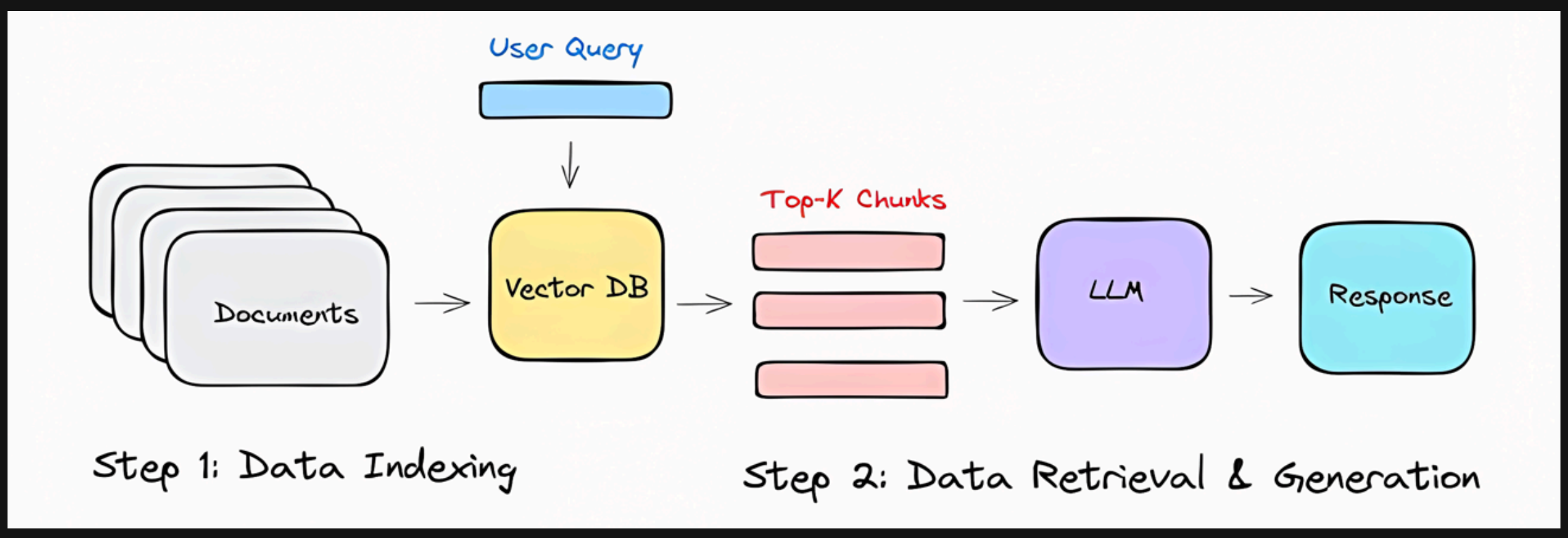


image source: medium.com/@drjulija/what-is-retrieval-augmented-generation-rag-938e4f6e03d1

To optimize the system we can apply **advanced techniques** in these areas

Pre-retrieval optimization

Optimize how you clean and prepare the data to create the chunks

Retrieval optimization

Improve queries

Post-retrieval optimization

Process results before sending them to the LLM

Evaluation of Generation

Evaluate RAG System responses

1

Ingestion

Load embeddings into the vector database

2

Retrieval

Query the vector database for relevant content

3

Generation

Use the retrieved context to answer user's query

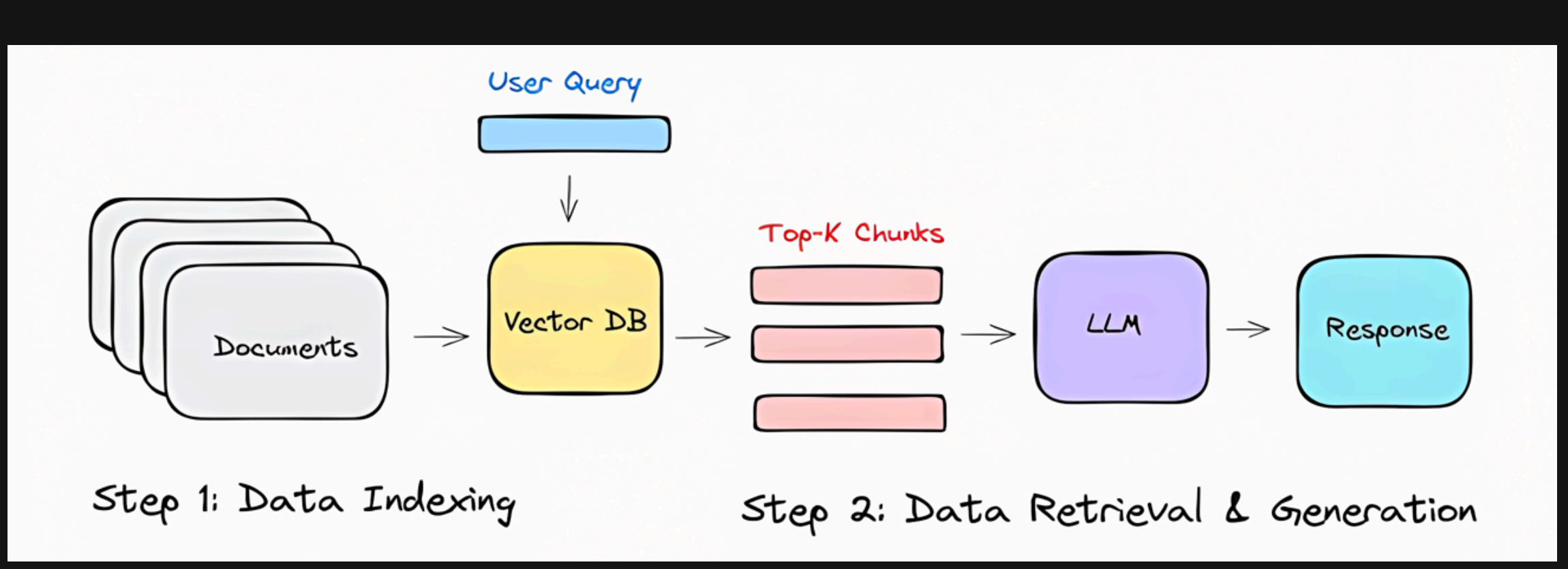


image source: medium.com/@drjulija/what-is-retrieval-augmented-generation-rag-938e4f6e03d1

We'll end up having a system like this

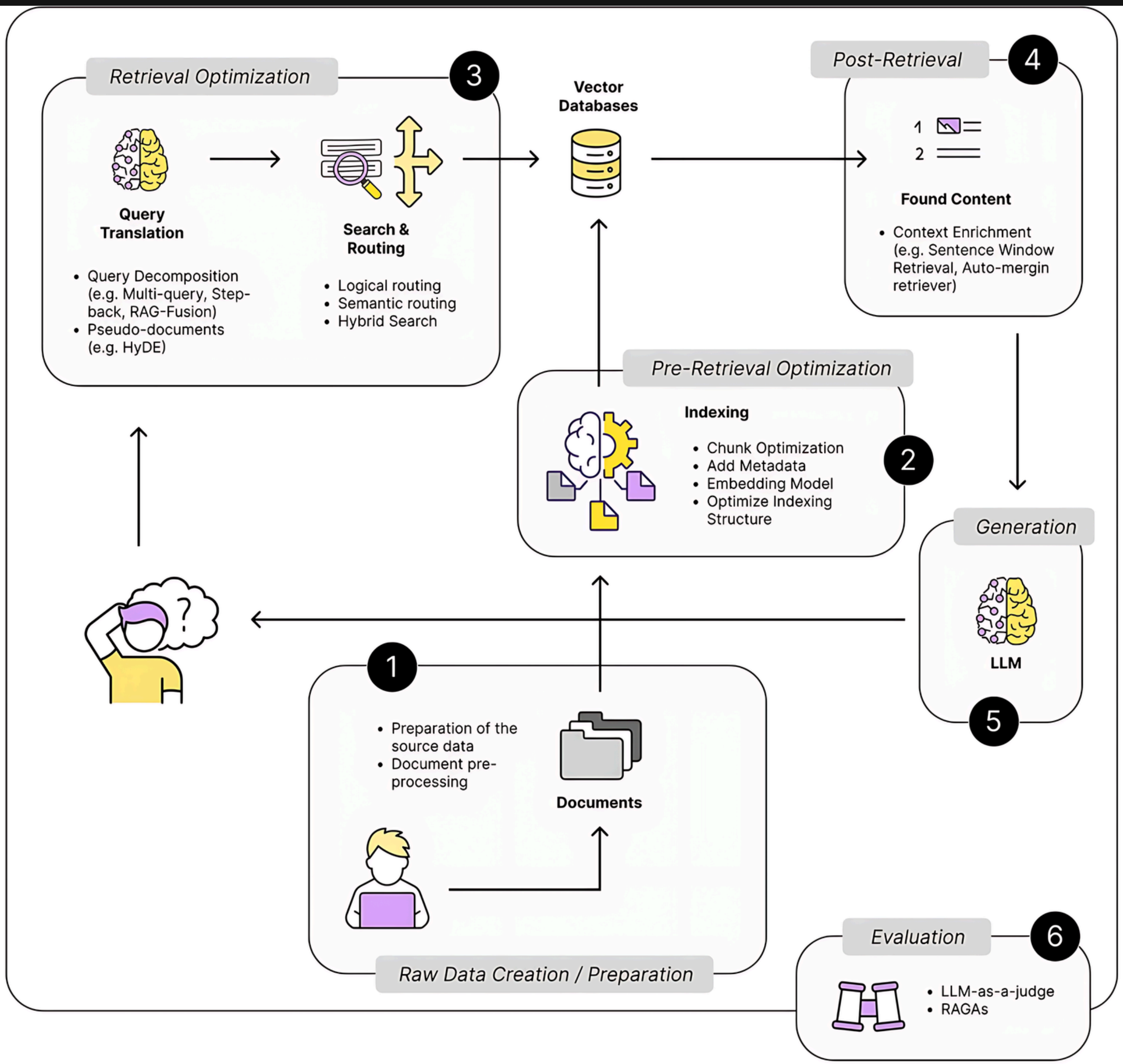
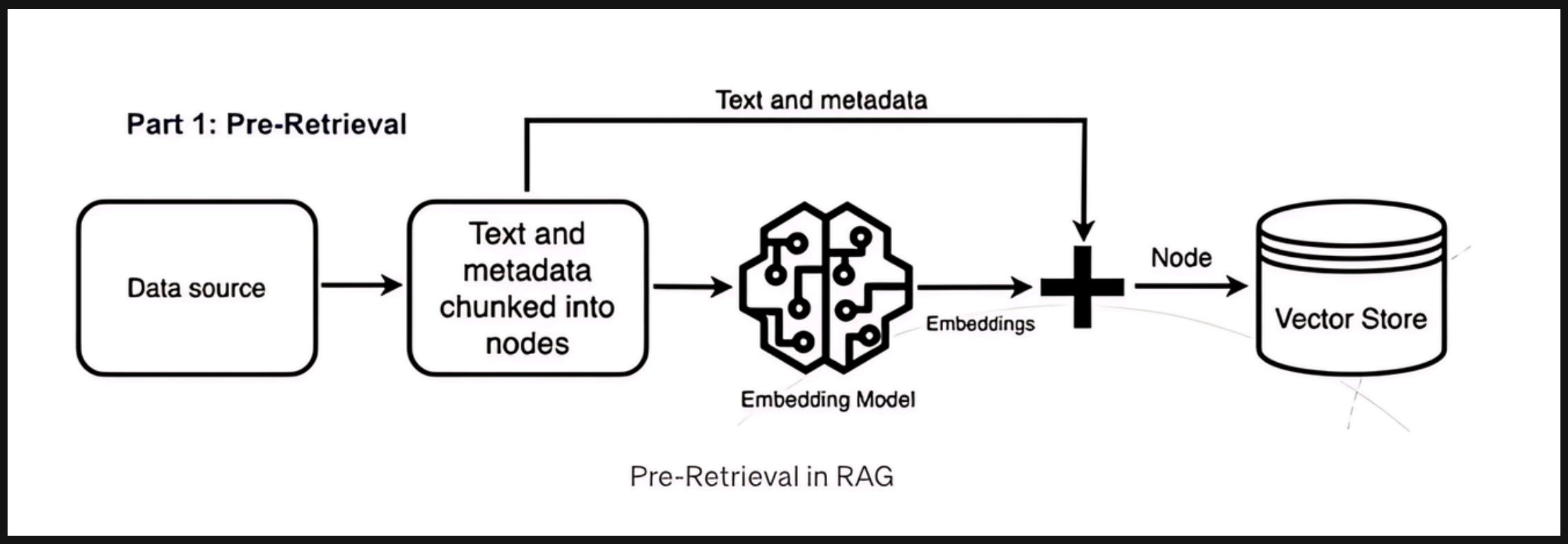


image source:towardsdatascience.com/17-advanced-rag-techniques-to-turn-your-rag-app-prototype-into-a-production-ready-solution-5a048e36cdc8

Let's look into these techniques one by one

Pre -Retrieval

Optimize how you clean and prepare the data to create the chunks



Pre-Retrieval in RAG

image source: https://luv-bansal.medium.com/advance-rag-improve-rag-performance-208ffad5bb6a

1 RAW Data Preparation

The problem

- LLMs struggle to fully understand context if text chunks are not self-explanatory.
- Domain-specific abbreviations and internal terms can confuse the model.

Solution

- **Ensure text chunks are self-explanatory by providing clear context**
- **Structure content to improve readability and understanding.**
- **Re-write technical documents to be clear and concise for both humans and LLMs.**

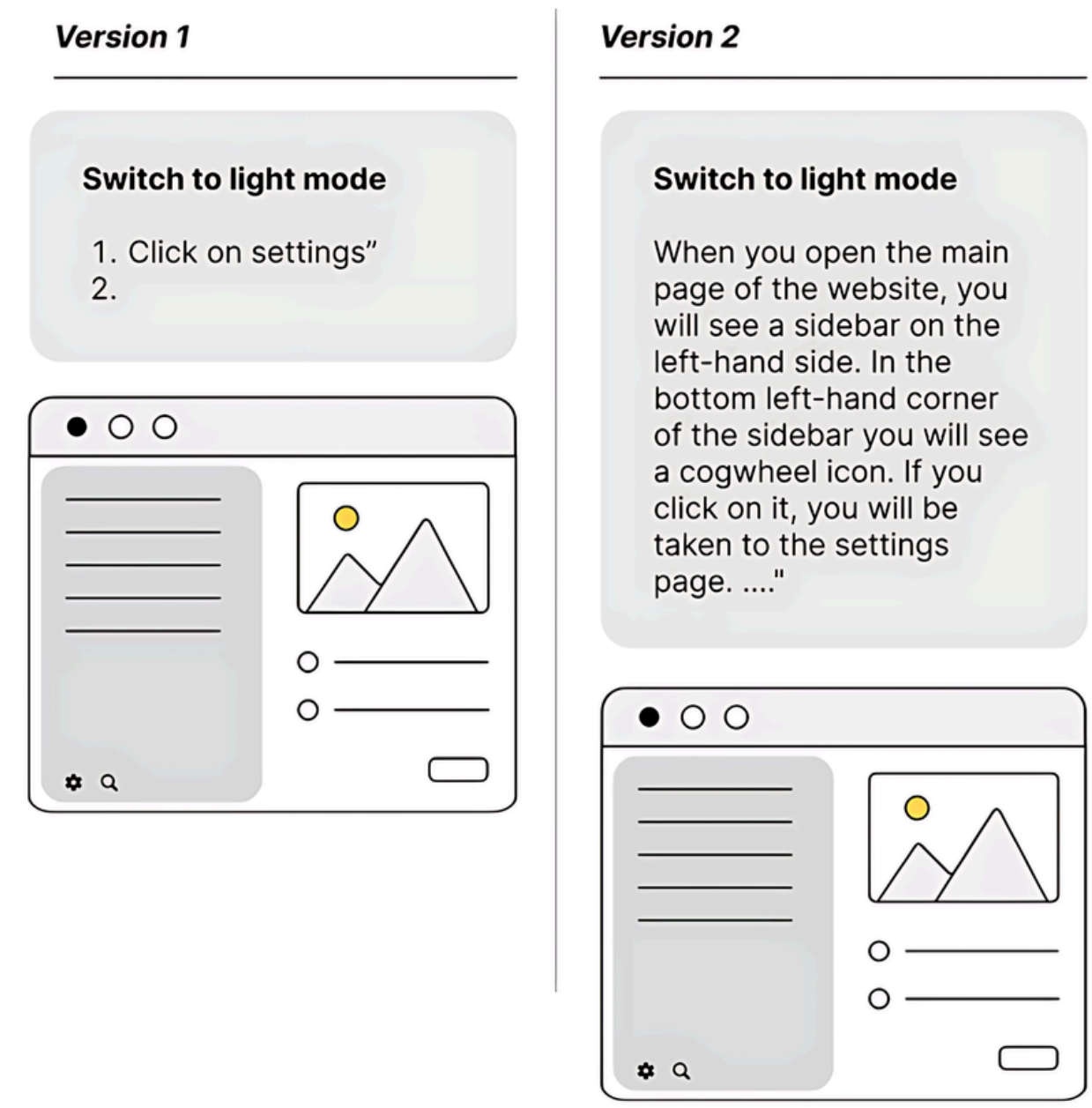


image source:towardsdatascience.com/17-advanced-rag-techniques-to-turn-your-rag-app-prototype-into-a-production-ready-solution-5a048e36cdc8

2 Indexing/Chunking Optimization

The problem

- Incorrect chunk sizes can lead to inefficient searches and poor performance.

- Embedding models have fixed input sequence lengths, limiting the token size of prompts.

Solution: Optimize Chunk Size

- Balance context and specificity for optimal performance.

Chunking Techniques For RAG

Technique	Usecase	Pros	Cons
Character splitter	Text	Versatile: Handles various separators Flexible: Adapts to different languages Cost-Effective: Does not require a ML model	Performance: May have increased computational load Complexity: Requires parameter tuning Sentence Interruption: May cut sentences midway
Recursive character splitter	Text, code	Versatile: Handles various separators Flexible: Adapts to different languages Cost-Effective: Does not require a ML model	Performance: Recursive nature may increase computational load Complexity: Requires parameter tuning Sentence Interruption: May cut sentences midway
Sentence splitter	Text	Considers Sentence Boundaries: Avoids cutting sentences prematurely Customizable: Parameters for stride and overlap Cost-Effective: Works with light sentence segmenter	Lack of Versatility: Limited to sentence-based chunks Overlap Issues: May lead to redundancy
Semantic splitter	Text, Chat	Contextual Grouping: Organizes text based on semantic similarity Overcomes Challen ges: Handles chunk size and overlap	Complexity: Requires similarity model and tuning Parameter Dependency: Relies on setting appropriate parameters Resource Intensive: Demands computational resources
Propositons	Text, Chat	Atomic Expression: Introduces novel retrieval unit (propositions) Distinct Factoids: Each proposition is self-contained Contextualization: Provides necessary context	Complexity: Requires LLM model Parameter Dependency: Relies on setting appropriate prompt Resource Intensive: Demands computational resources

Galileo

3 Improve data quality

The problem
‘*Garbage in, garbage out*’

- Abbreviations, technical terms, and internal jargon can hinder the LLM's ability to understand and process the text.
- Lack of context can lead to misinterpretation and inaccuracies.

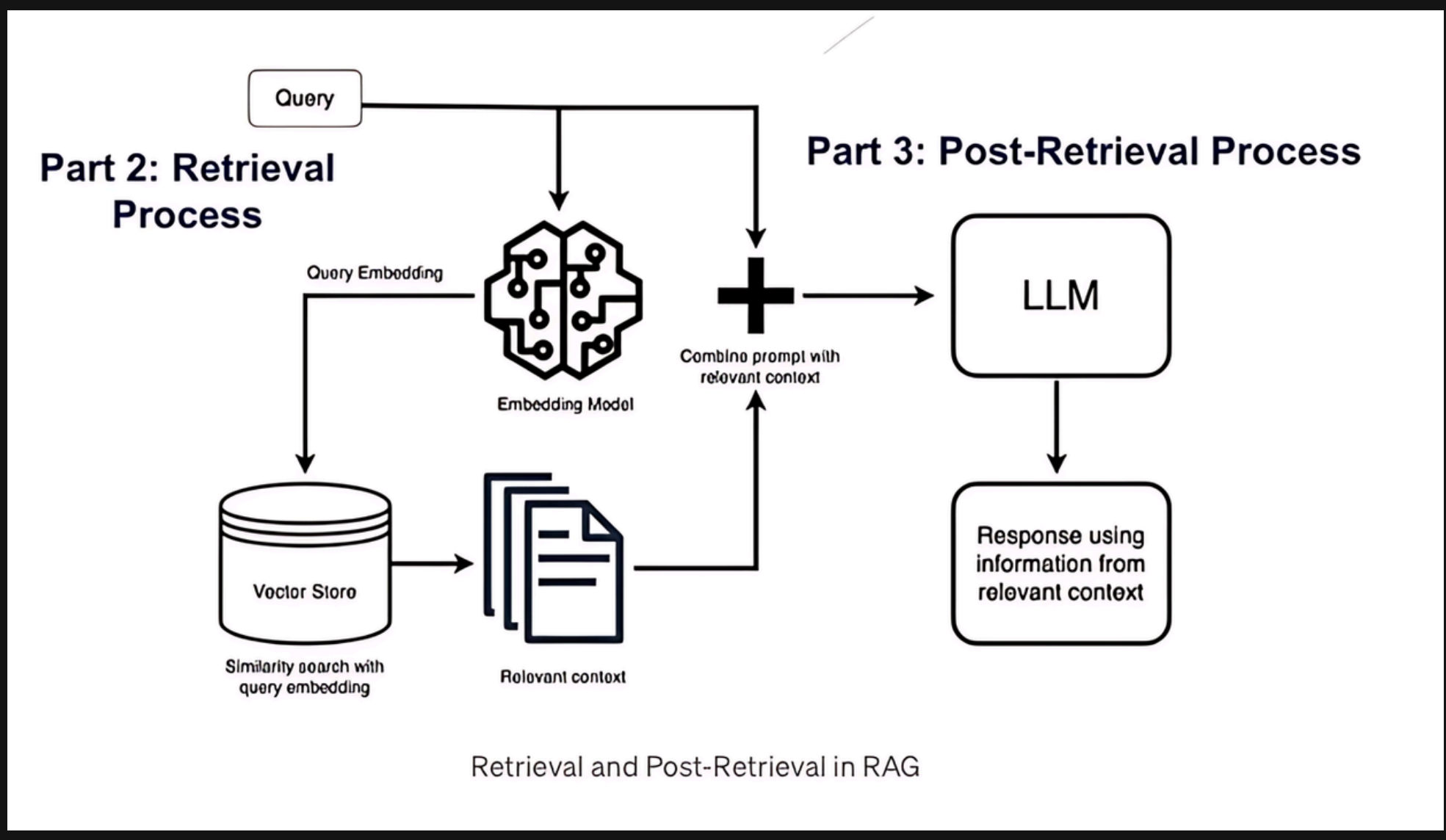
Solutions

- **Data Cleansing**
  - Replace abbreviations with full terms using an abbreviation translation table.
  - Remove noise data, and HTML tags.

- **Add metadata**
  - Use metadata such as concept and level tags, dates and purposes to improve the quality of indexed data.

Retrieval

Improve queries for higher relevance



Retrieval and Post-Retrieval in RAG

image source: https://luv-bansal.medium.com/advance-rag-improve-rag-performance-208ffad5bb6a

4 Retrieval optimization

The problem

- Simple queries might not effectively retrieve the most relevant content.
- Diverse user queries require flexible and robust retrieval strategies.

Solutions

- **Query Expansion (HyDE)**
  - Use LLM to generate hypothetical answers before performing similarity search, increasing the likelihood of finding relevant information.
- **Multiple System Prompts**
  - Generate multiple prompts with slight variations to gather diverse responses, improving the chances of retrieving relevant content.
- **Query Routing**
  - Direct queries to the most relevant data collections based on initial LLM analysis, ensuring targeted and accurate retrieval.

5 Hybrid Search

The problem

- Relying solely on either vector search or keyword search can lead to incomplete or irrelevant results.

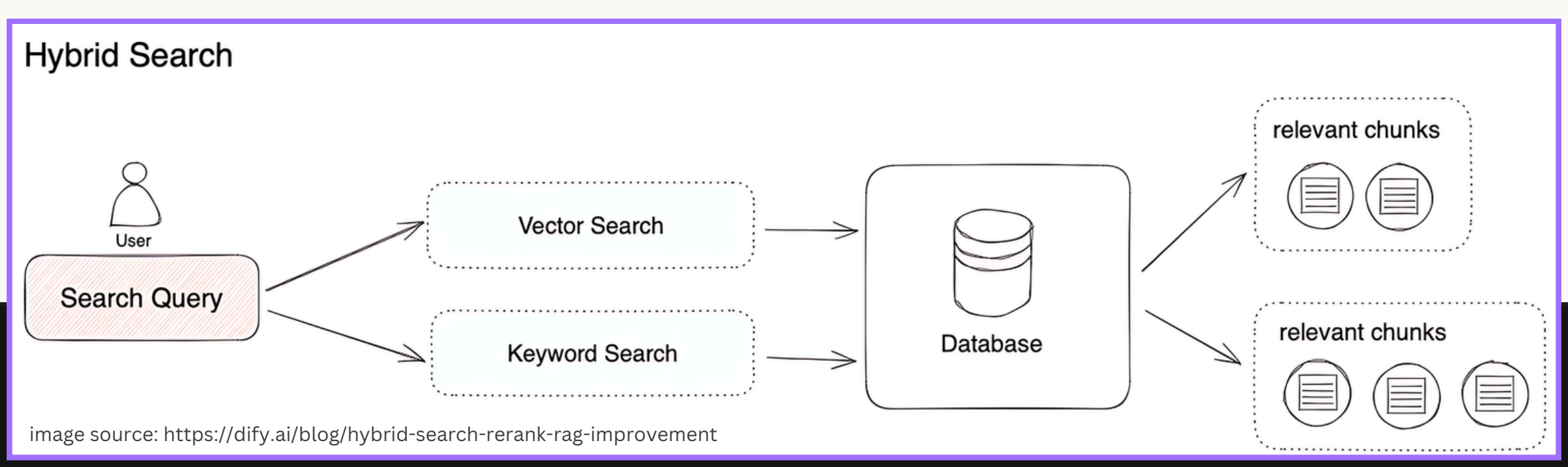


image source: https://dify.ai/blog/hybrid-search-rerank-rag-improvement

Solutions

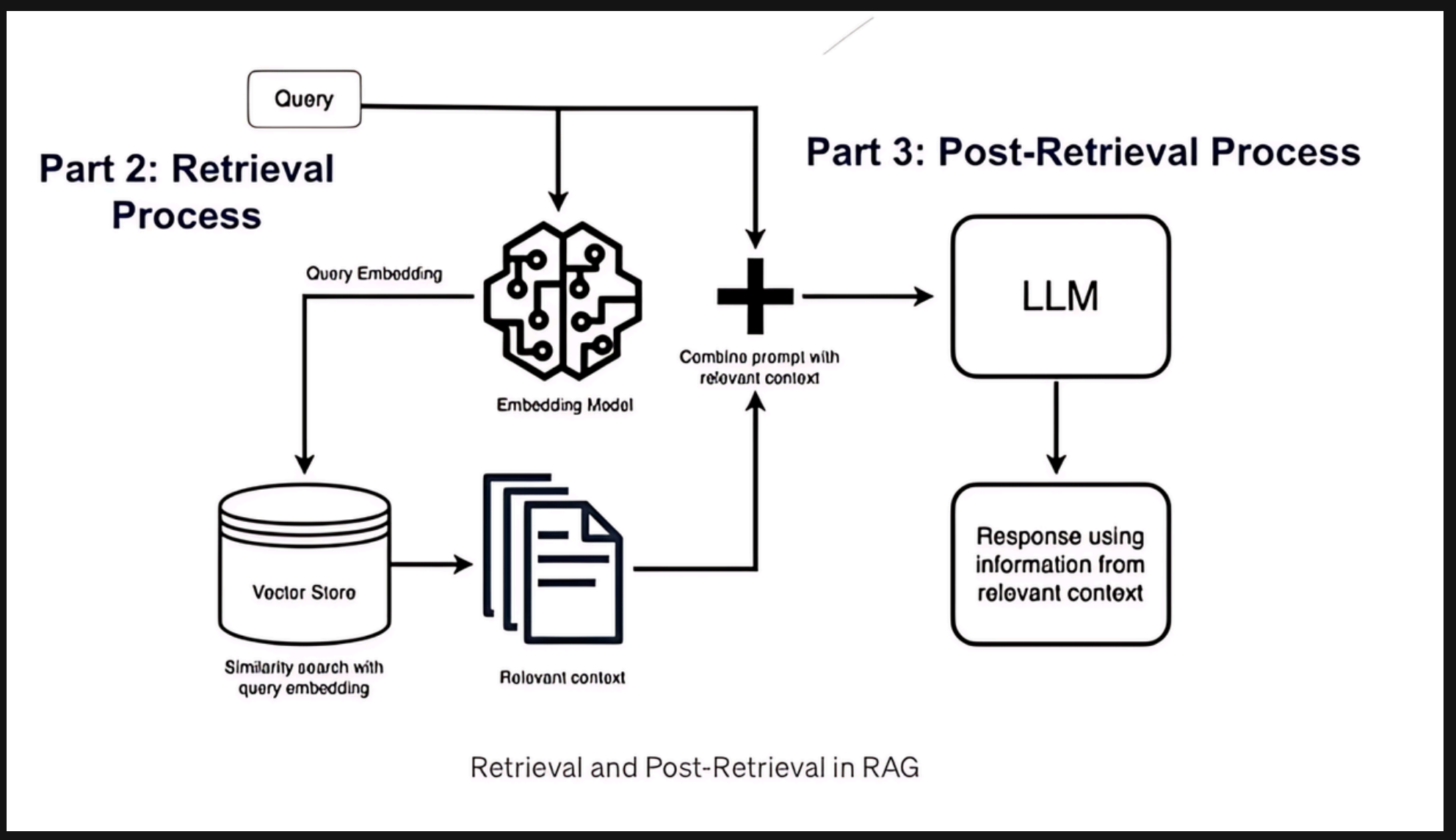
Combining Searches

- Perform both vector (semantic) search and lexical (keyword) search to leverage the strengths of both methods.

- Consolidate and rank the results from both searches to provide the most accurate and relevant information.

Post-Retrieval

Pre-process results before sending them to the LLM



Retrieval and Post-Retrieval in RAG

image source: https://luv-bansal.medium.com/advance-rag-improve-rag-performance-208ffad5bb6a

6 Context Enrichment

The problem

- Small text chunks might lack sufficient context, leading to incomplete or less accurate answers.
- Providing additional context can enhance the LLM's understanding and response quality.

Solutions

- **Sentence Window Retrieval**
  - Add the k-sentences before and after the best-matching text chunk to provide more context.
- **Auto-Merging Retriever**
  - Use parent-child node relationships to enrich the retrieved chunks with related information.

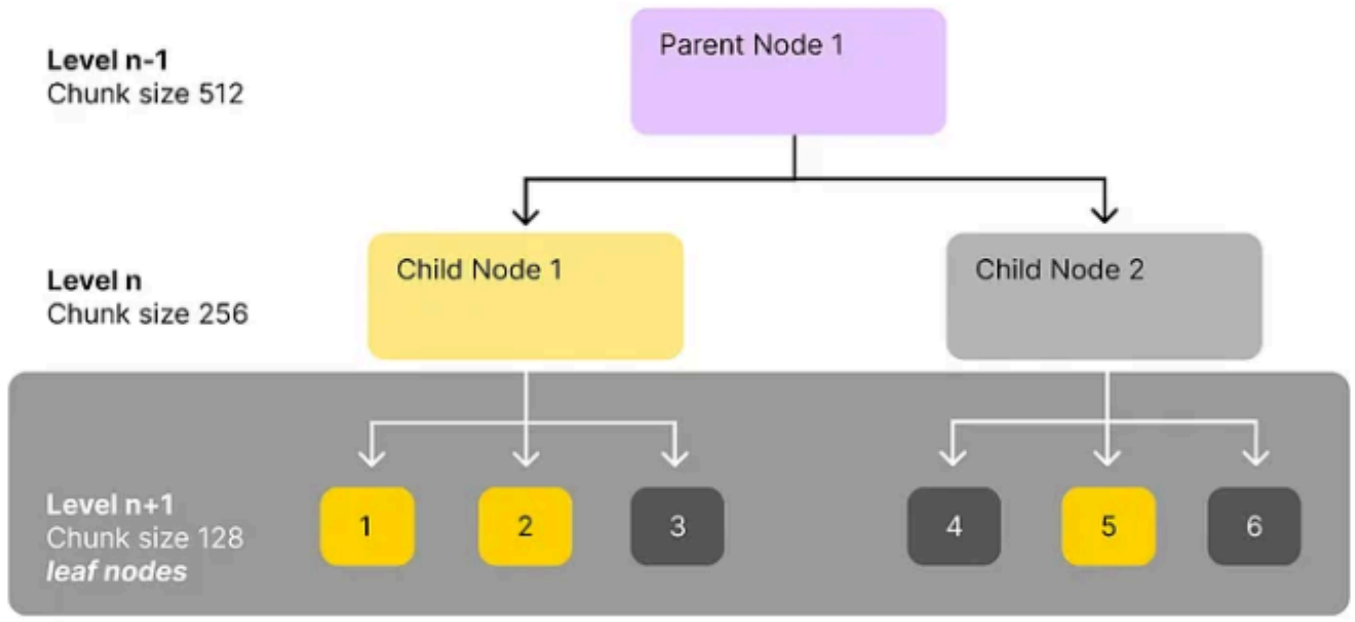


image source:towardsdatascience.com/17-advanced-rag-techniques-to-turn-your-rag-app-prototype-into-a-production-ready-solution-5a048e36cdc8

7 Re Ranking

The problem

- A high score in vector similarity search doesn't always guarantee the highest relevance, leading to less effective responses.

Solution

- **Re-Ranke retrieval result**
  - Rerank retrieval results before sending them to the LLM to significantly improve RAG performance.
  - Increase the similarity_top_k in the query engine to retrieve more context passages, then reduce to top_n after reranking for optimal results.

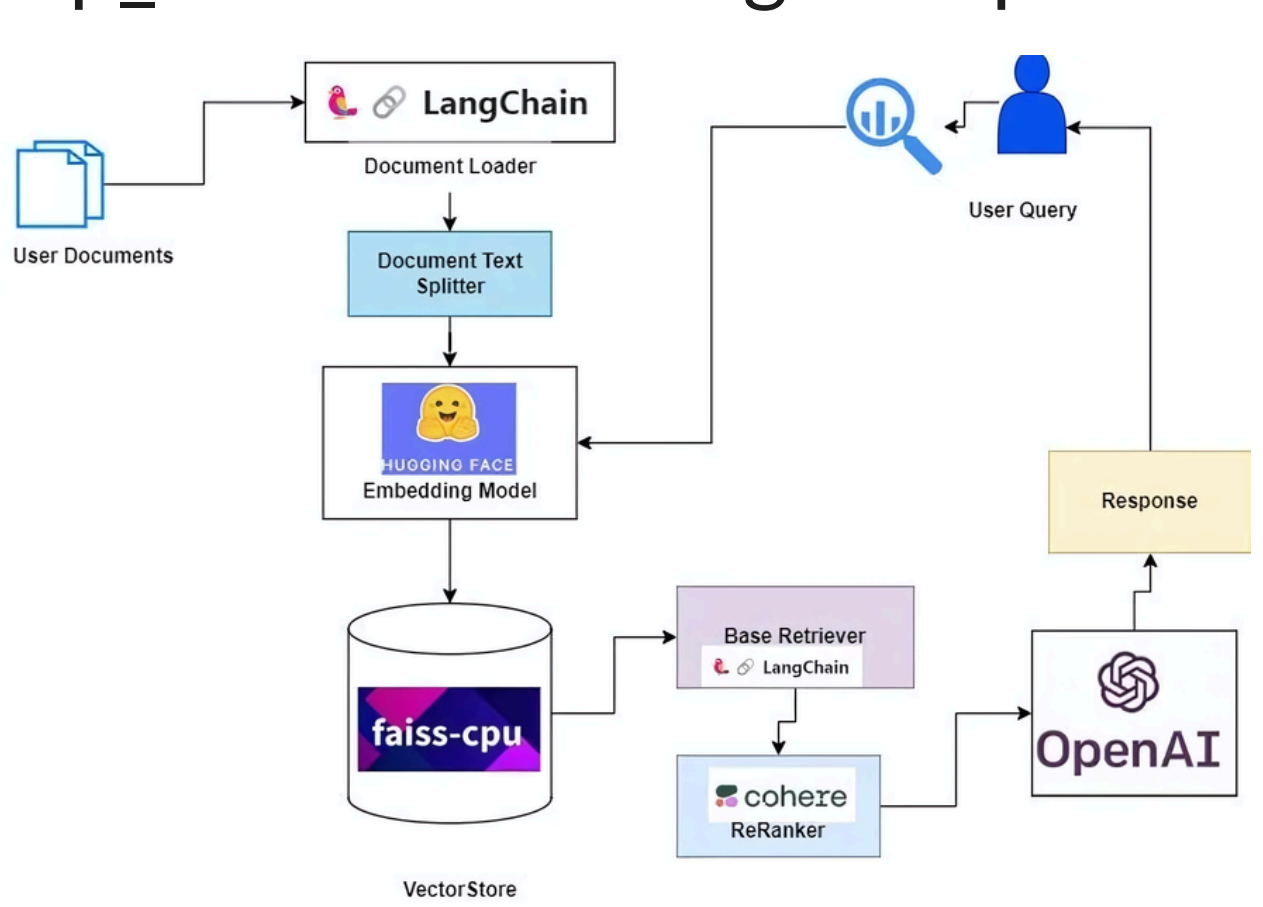


image source: https://luv-bansal.medium.com/advance-rag-improve-rag-performance-208ffad5bb6a

Generation

Use the retrieved context
to solve the user's query

8 Implement Agents

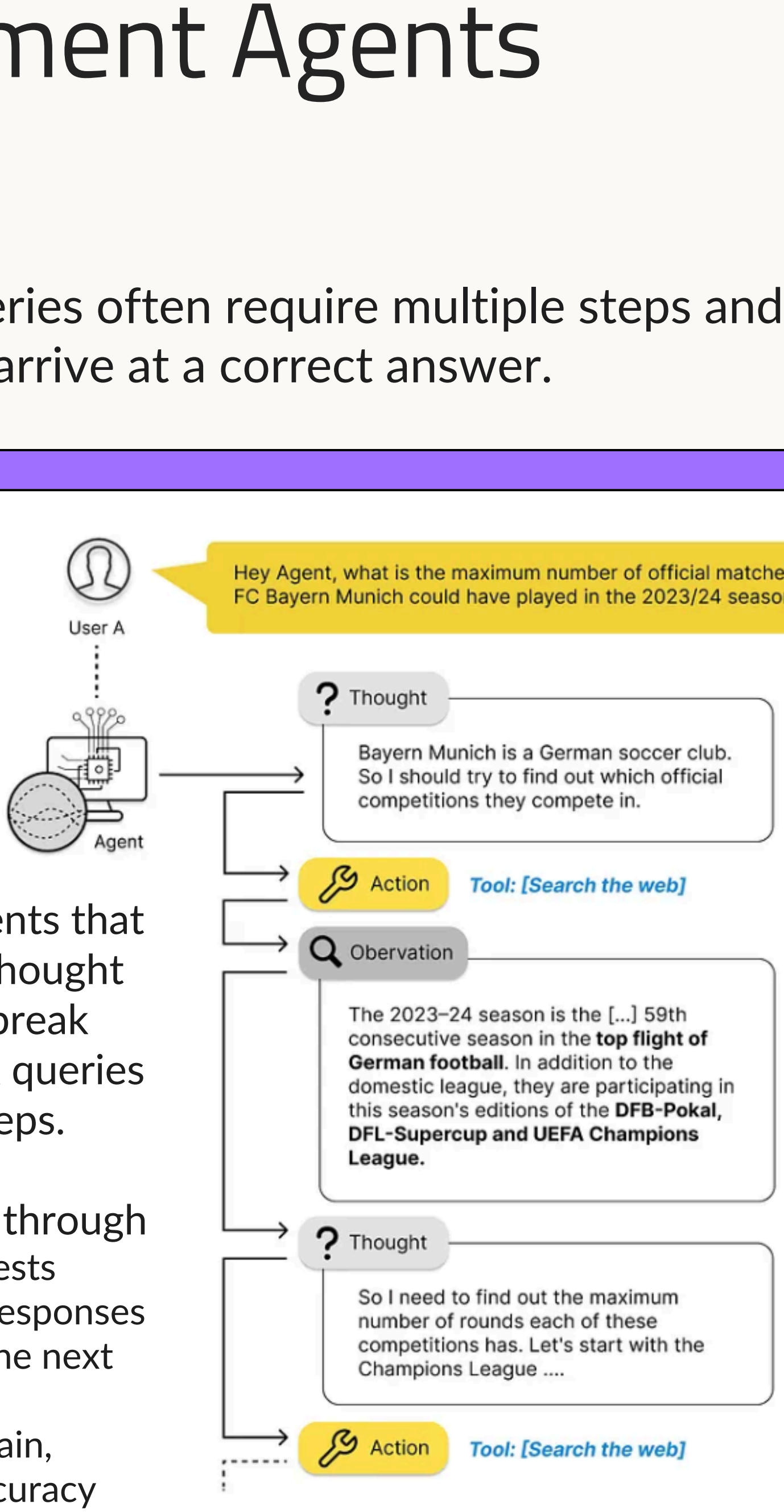
The problem

Complex queries often require multiple steps and decisions to arrive at a correct answer.

Solution

- **Agents**
  - Implement agents that use "chain of thought reasoning" to break down complex queries into simpler steps.

  - Agents iterate through
    - sending requests
    - interpreting responses
    - deciding on the next step,
    - and acting again, improving accuracy over one-shot prompting



How do agents work? — Image by the author

image source:towardsdatascience.com/17-advanced-rag-techniques-to-turn-your-rag-app-prototype-into-a-production-ready-solution-5a048e36cdc8

Evaluation

Implement frameworks to evaluate the RAG systems

9 Evaluating RAG Systems

The problem

- Evaluating the performance of RAG systems is complex due to the ambiguity of language and the multifaceted nature of the tasks.
- Regular human evaluation is impractical for continuous improvement.

Solutions

1. LLM-as-a-Judge

- Use LLMs to evaluate the quality of responses based on predefined criteria.
- Set up critique agents to automatically assess responses for attributes like relevance, accuracy, and professionalism.

2. Implement RAGAS evaluation Framework

- Use metrics like faithfulness, context precision, and semantic similarity to assess both individual components and the system as a whole.

We tried many of these techniques, and no single technique works all the time.

It involves a lot of experimentation, and it's good to have these ideas as an inspiration

I highly suggest to check the resources bellow for a deep dive into the techniques

References:

17 (Advanced) RAG Techniques to Turn Your LLM App Prototype into a Production-Ready Solution | by Dominik Polzer |

The 4 Advanced RAG Algorithms You Must Know to Implement

10 Ways to Improve the Performance of Retrieval Augmented Generation Systems | by Matt Ambrogi

Mastering RAG: Advanced Chunking Techniques for LLM Applications - Galileo

Advance RAG- Improve RAG performance

I'm Joanna from Labsbit.ai

A Gen-AI Product Development Company

Reach out to learn more about how we can help you build your RAG app