Smarter Retrieval, Better Generation: Improving RAG Systems

David S. Batista

PyCon Lithuania - Vilnius, April 2025

About Me

Lisbon, Portugal

About Me

Lisbon, Portugal

Berlin, Germany

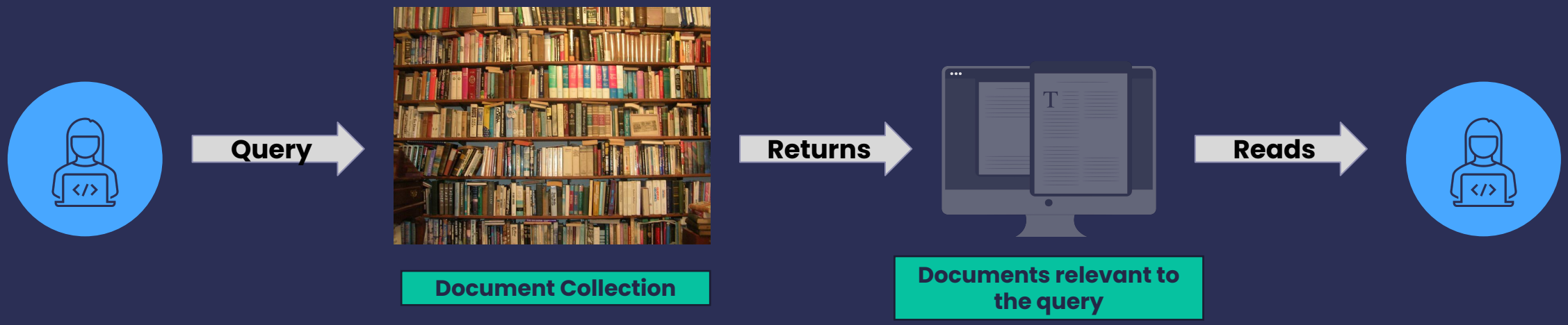
About Me

haystack
by deepset

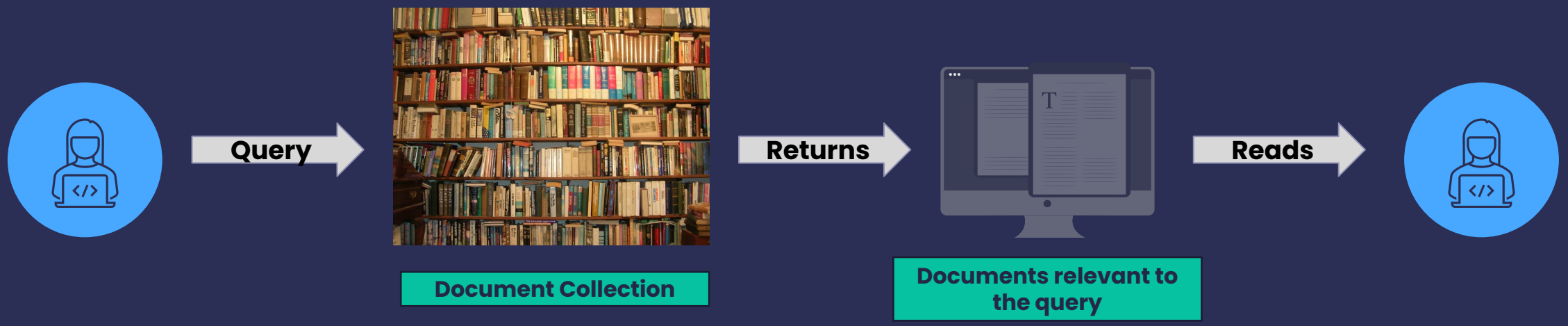
open source framework for building LLM applications, retrieval-augmented generative pipelines search systems that work over large document collections

Berlin, Germany

From classic Information Systems to RAG

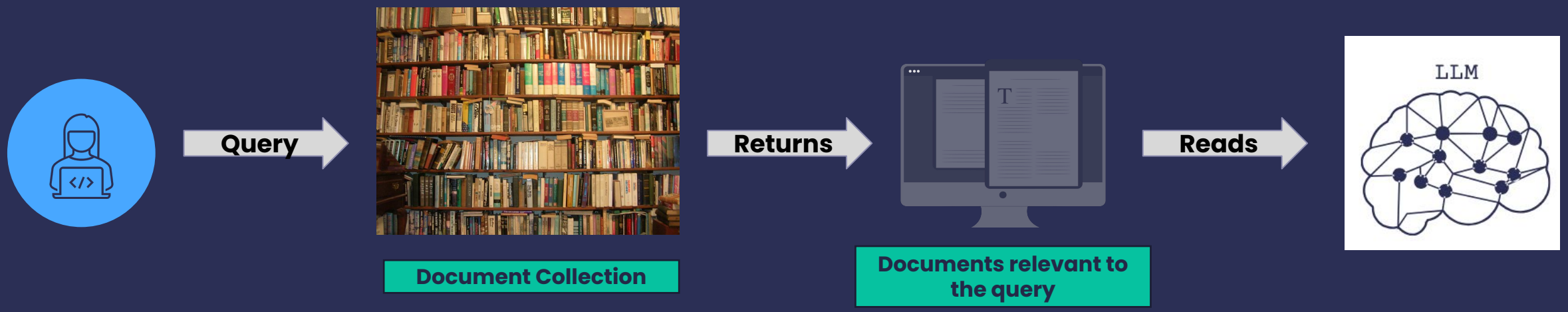


From classic Information Systems to RAG



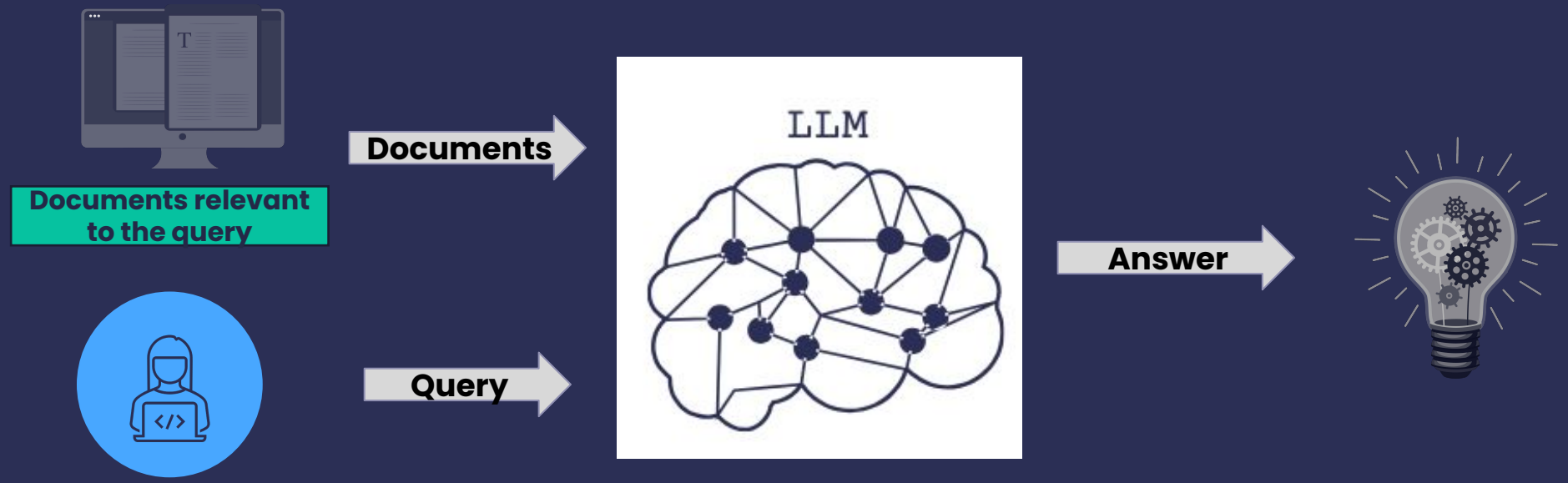
- Return a list of documents or snippets, requiring users to read through multiple results to find the information they need
- A complex or nuanced query requires a deeper understanding of the context and relationships between different pieces of information

From classic Information Systems to RAG

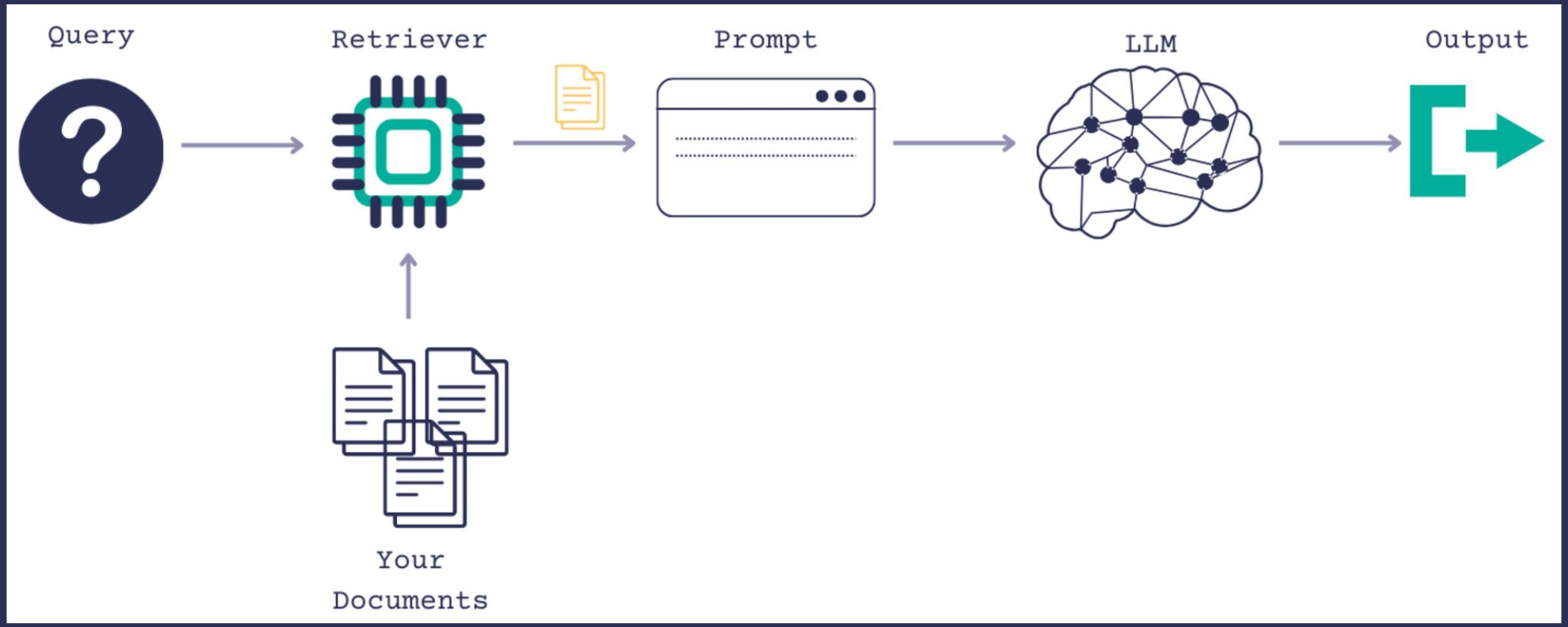


- What if, instead the user sifting through the results, we build a prompt composed by retrieved snippets together with the query and feed it to an LLM?

From classic Information Systems to RAG

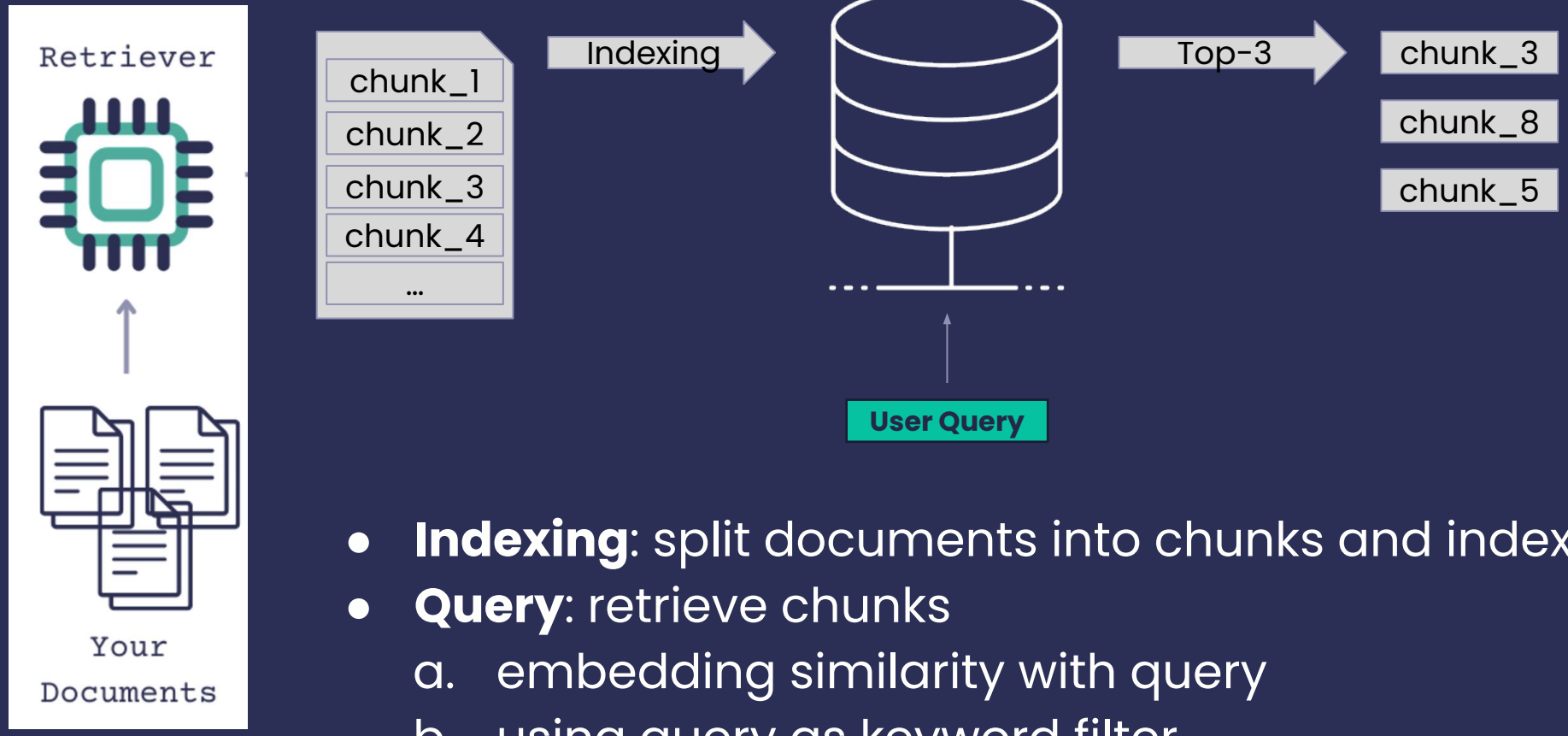


RAG - Retrieval Augmented Generation



RAG - Retrieval Step

Motivation - Baseline Retrieval



- **Indexing**: split documents into chunks and index in a vector db
- **Query**: retrieve chunks
  a. embedding similarity with query
  b. using query as keyword filter
- **Ranking**: rank by similarity with the query

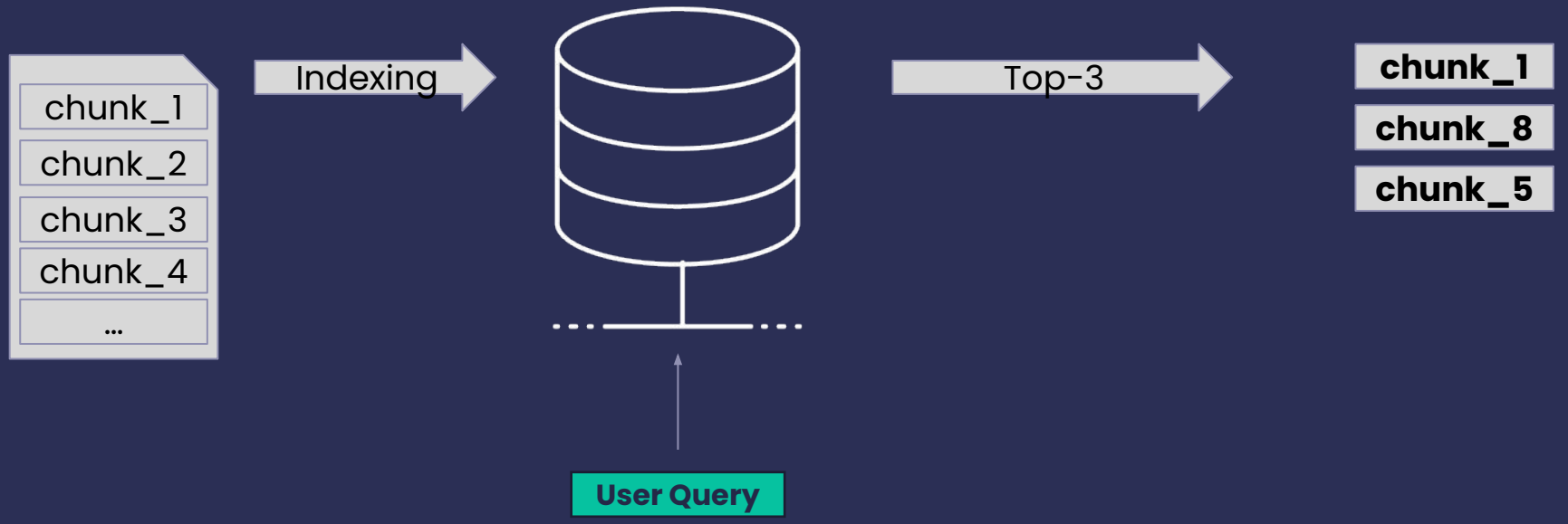
Outline

1. Classic Retrieval Techniques
2. LLM-based Retrieval Techniques
3. Comparative Summary
4. Experiment

Classical Techniques

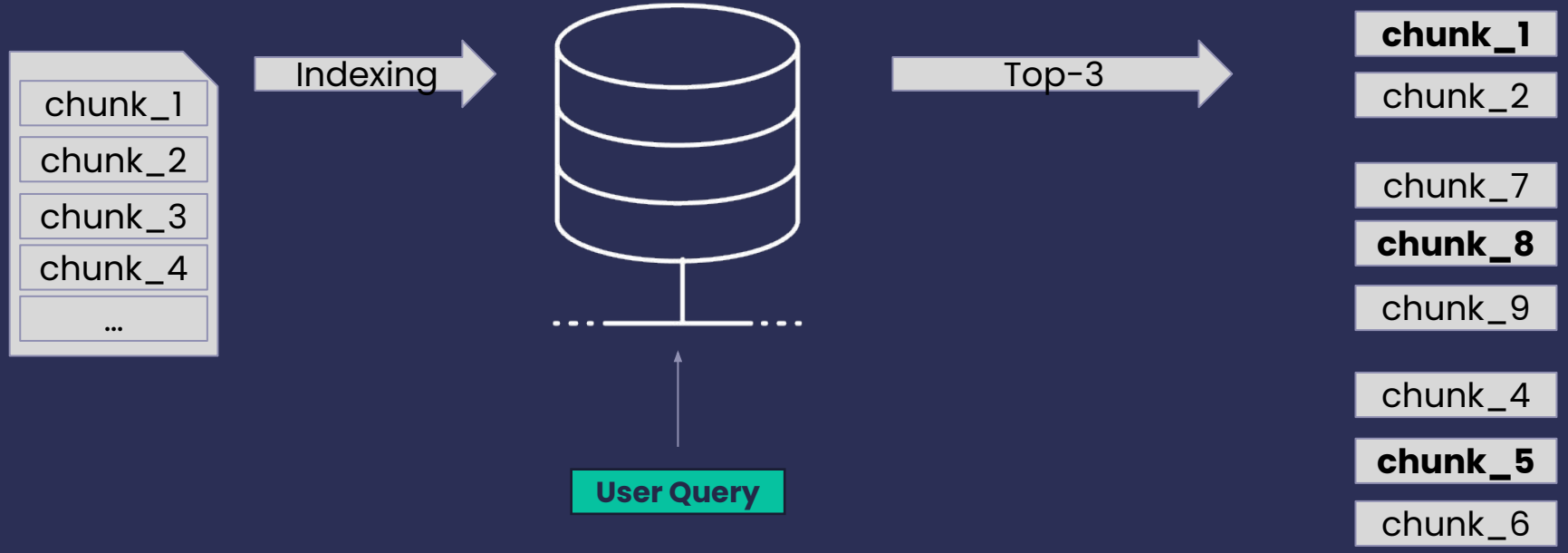
- Sentence-Window Retrieval
- Auto-Merging Retrieval
- Maximum Marginal Relevance
- Hybrid Retrieval (with/ Reciprocal Rank fusion)

Sentence-Window Retrieval



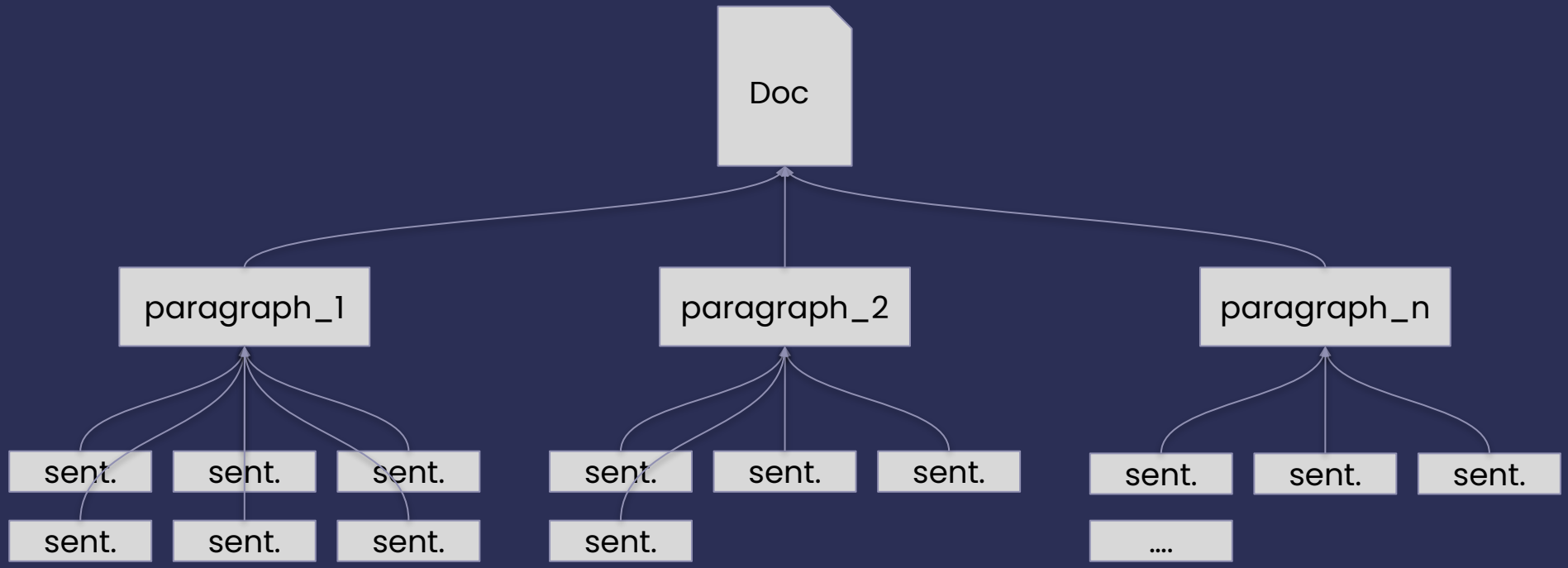
- Retrieve the chunks before and after the matching chunk

Sentence-Window Retrieval



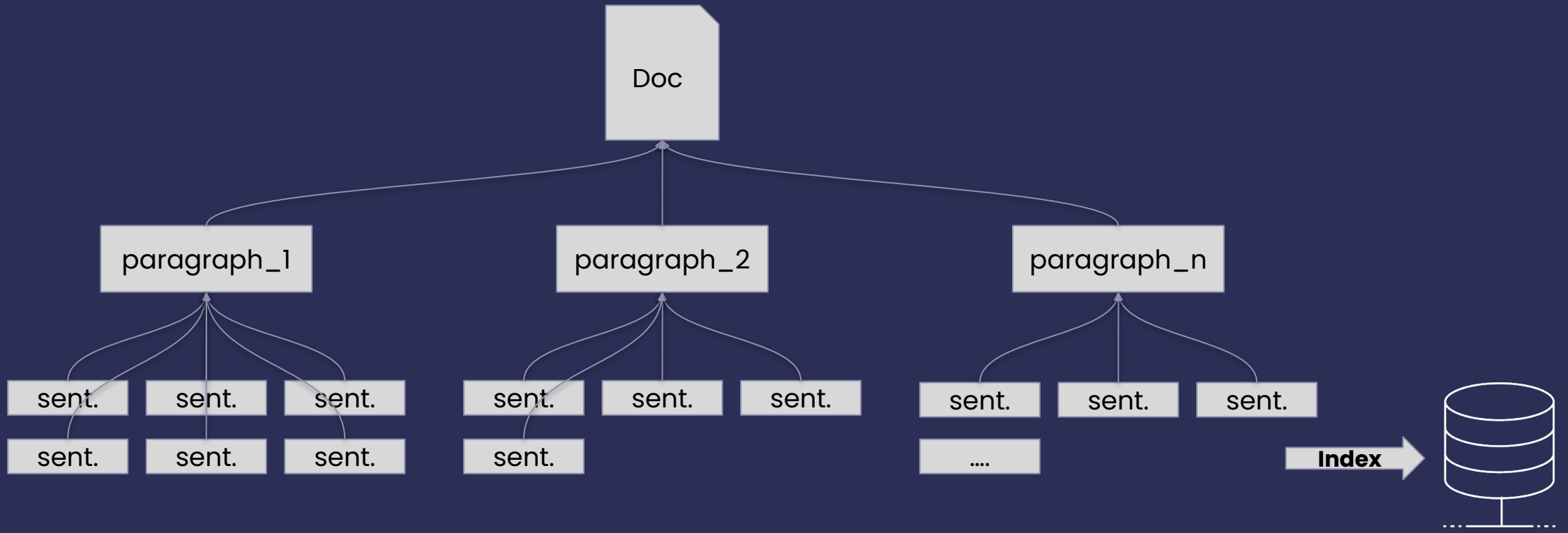
- Retrieve the chunks before and after the matching chunk
- A simple way to gather more context
- Indexing needs to preserve the order of the chunks

Auto-Merging Retrieval



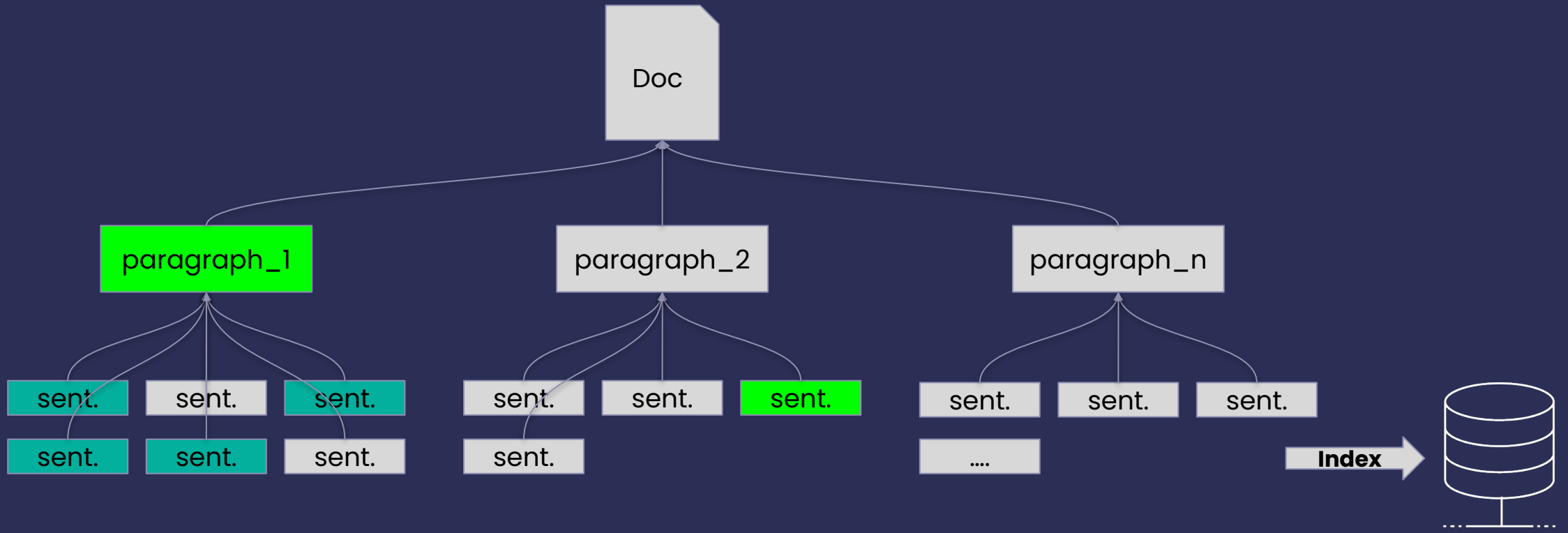
- Transform documents into an Hierarchical Tree structure

Auto-Merging Retrieval



- Transform documents into an Hierarchical Tree structure
- Children chunks/sentences are index and used for retrieval

Auto-Merging Retrieval



- With a threshold of 0.5
  - The paragraph_1 is returned, instead of 4 sentences
  - Plus, the one sentence from paragraph_2
- A whole paragraph might be more informative than individual chunks

Maximum Marginal Relevance (MMR)

- Classical retrieval ranks the retrieved documents by relevance similarity to the user query
- What about scenarios with a high number of relevant documents, but also highly redundant or containing partially or fully duplicative information?
- We need to consider how novel is a document compared to the already retrieved docs

Maximum Marginal Relevance (MMR)

$$\text{MMR} = \arg\max_{d_i \in D \setminus R} [\lambda \cdot \text{Sim}_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in R} \text{Sim}_2(d_i, d_j)]$$

Each retrieved document is scored

Maximum Marginal Relevance (MMR)

$$\lambda \cdot \text{Sim}_1(d_i, q)$$

Each retrieved document is scored:

- Similarity between a candidate document and the query

Maximum Marginal Relevance (MMR)

$$- (1 - \lambda) \cdot \max_{d_j \in R} \text{Sim}_2(d_i, d_j)]$$

Each retrieved document is scored:

- Find maximum similarity between a candidate document and any previously selected document.
- Maximize the similarity to already selected documents and then subtracting it - penalize documents that are too similar to what's already been selected.

Maximum Marginal Relevance (MMR)

$$[\lambda \cdot \mathrm{Sim}_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in R} \mathrm{Sim}_2(d_i, d_j)]$$

Each retrieved document is scored:

- Similarity between the candidate document and the query

- Find maximum similarity between a candidate document and any previously selected document.
- Maximize the similarity to already selected documents and then subtracting it - penalize documents that are too similar to what's already been selected.

- **λ** balances between these two terms

Maximum Marginal Relevance (MMR)

$$\text{MMR} = \arg\max_{d_i \in D \setminus R} [\lambda \cdot \text{Sim}_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in R} \text{Sim}_2(d_i, d_j)]$$

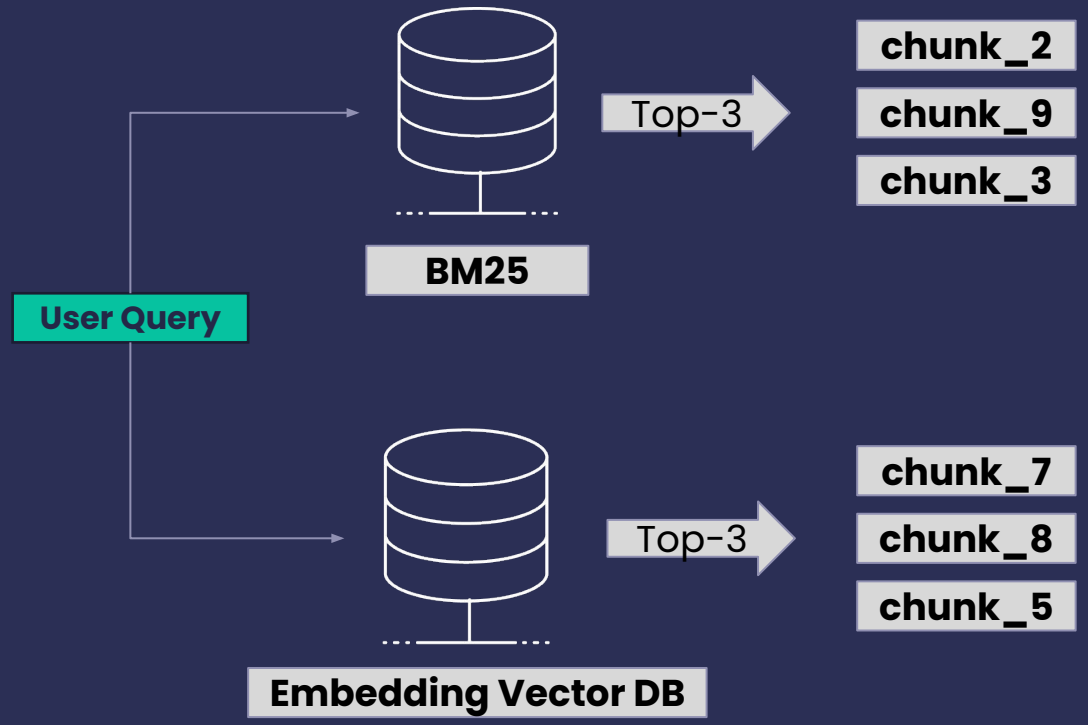
Each retrieved document is scored:

- Similarity between the candidate document and the query
- Find maximum similarity between the candidate document and any previously selected document. By maximizing the similarity to already selected documents and then subtracting it, we penalize documents that are too similar to what's already been selected.
- **λ** balances between these two terms

Hybrid Retrieval + Reranking

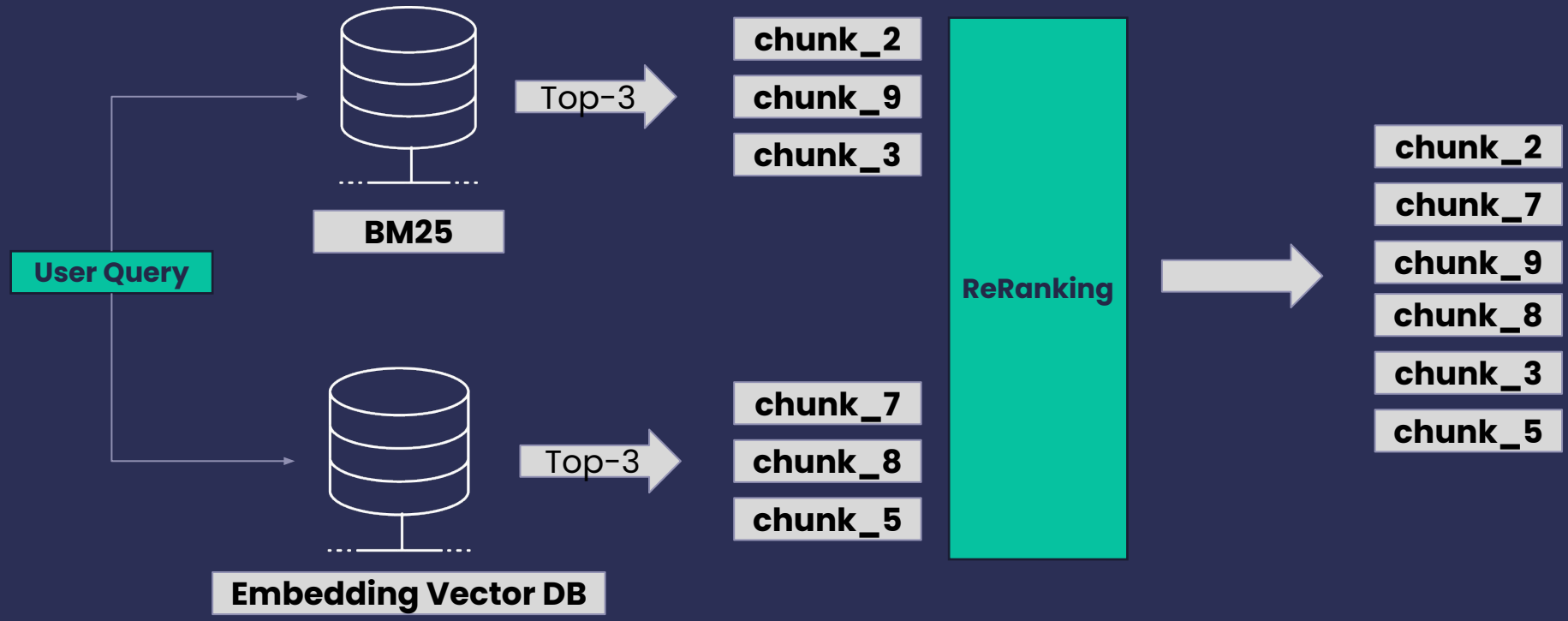
- Combines multiple search techniques
- keyword-based (BM25)

Hybrid Retrieval + Reranking



- Combines multiple search techniques
- keyword-based (BM25) and semantic-based (embedding vector)

Hybrid Retrieval + Reranking



- Combines multiple search techniques
- keyword-based (BM25) and semantic-based (embedding vector)
- Rank-merge results

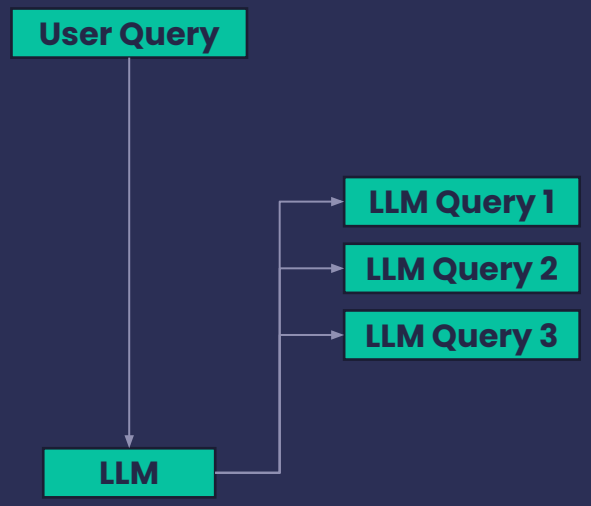
Classical Techniques

- ~~Sentence-Window Retrieval~~
- ~~Auto-Merging Retrieval~~
- ~~Maximum Marginal Relevance~~
- ~~Hybrid Retrieval (with/ Reciprocal Rank fusion)~~

LLM-based Techniques

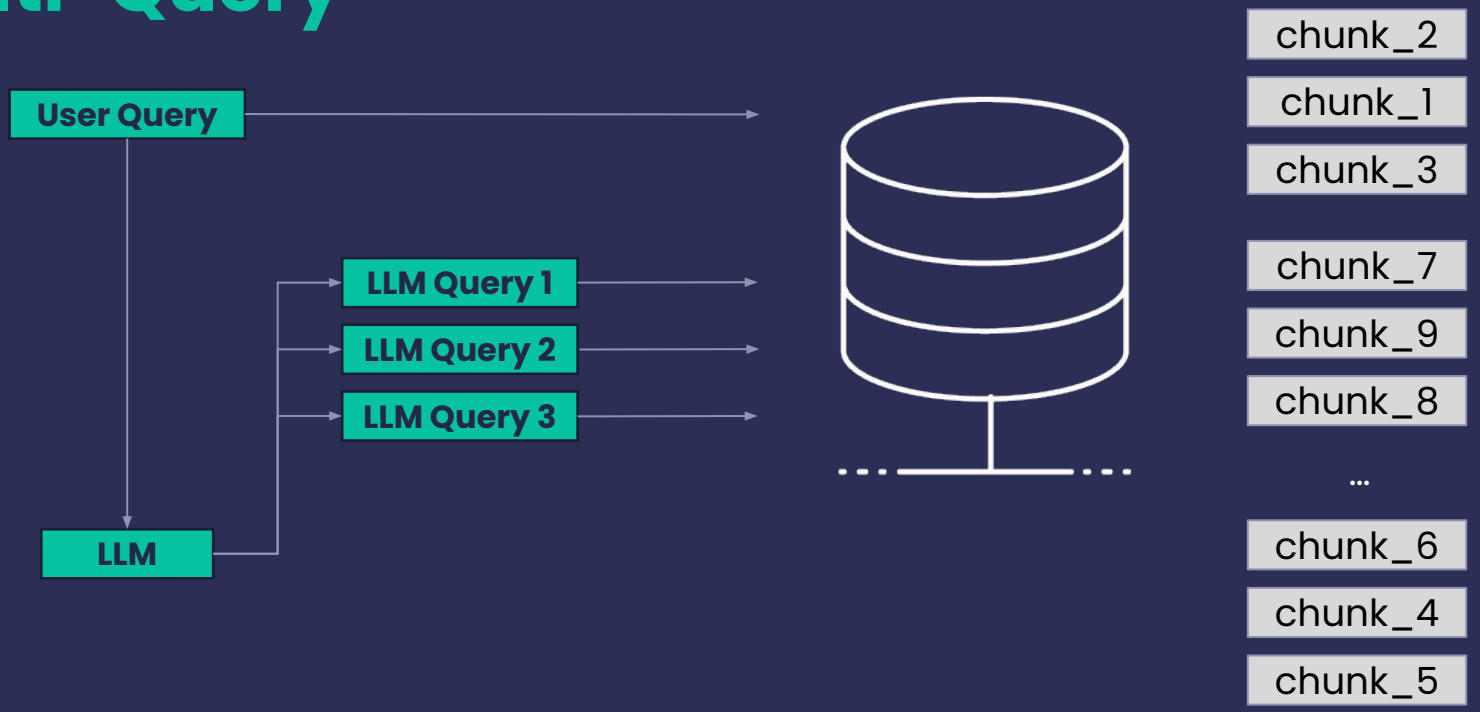
- Multi-Query
- Hypothetical Document Embeddings - HyDE
- Document Summary Indexing

Multi-Query



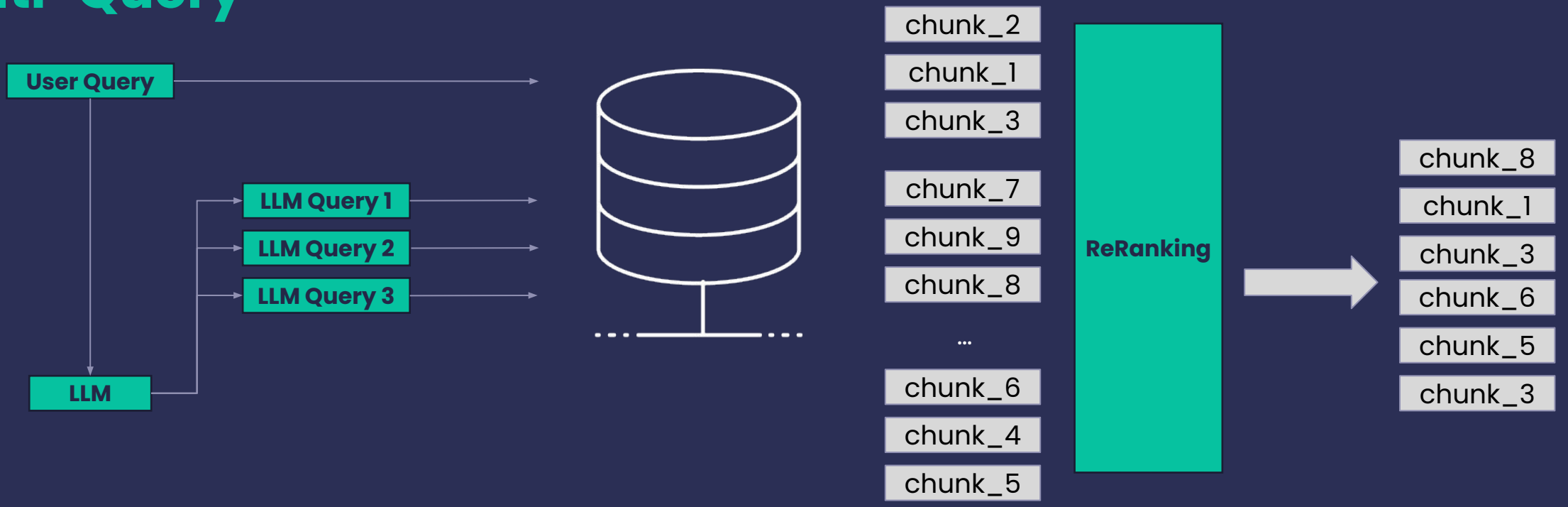
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions

Multi-Query



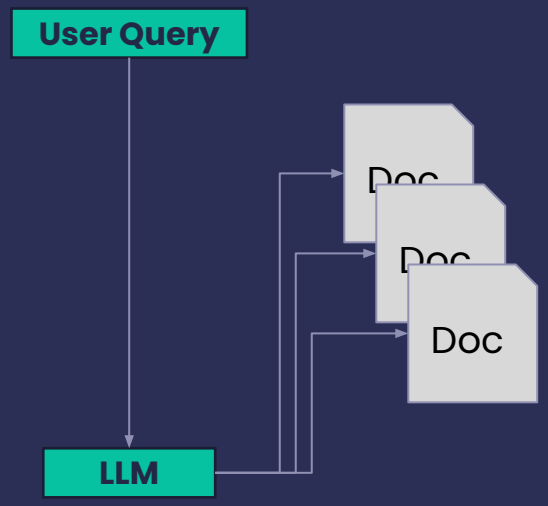
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions
- Each new query is used for an individual retrieval processes

Multi-Query



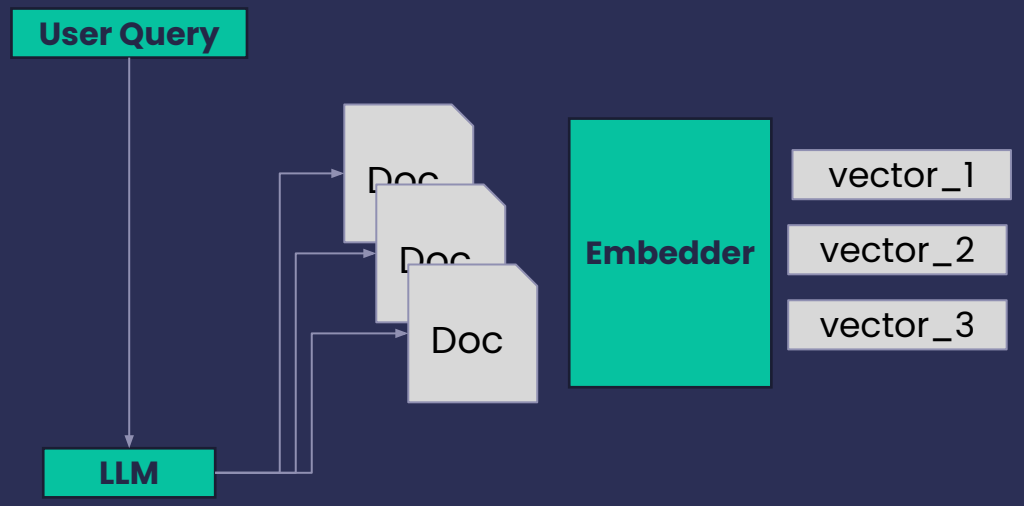
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions
- Each new query is used for an individual retrieval processes
- Re-ranking process over all retrieved chunks

Hypothetical Document Embeddings – HyDE



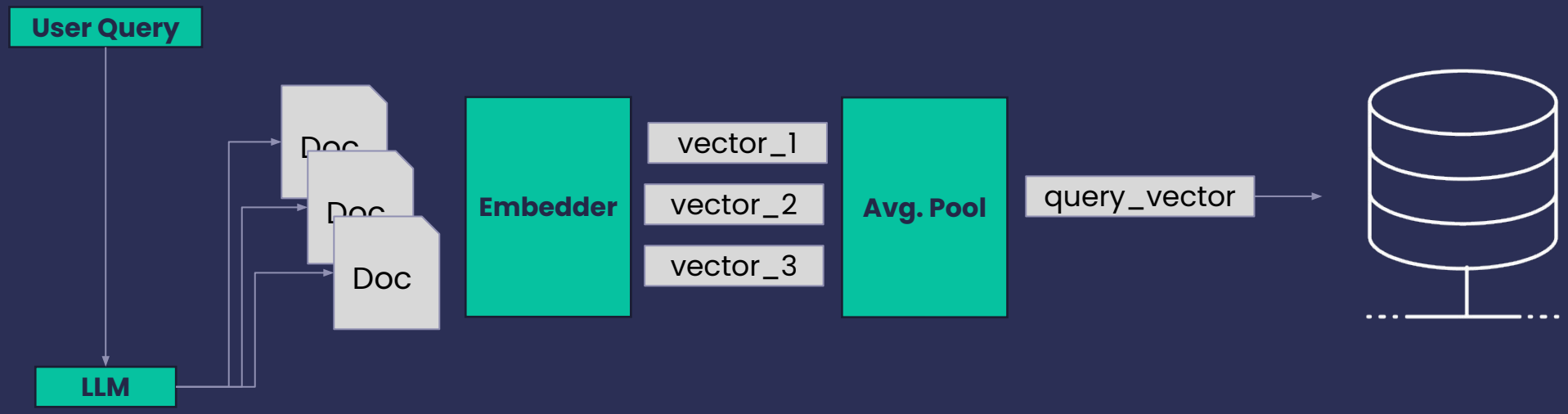
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query

Hypothetical Document Embeddings – HyDE



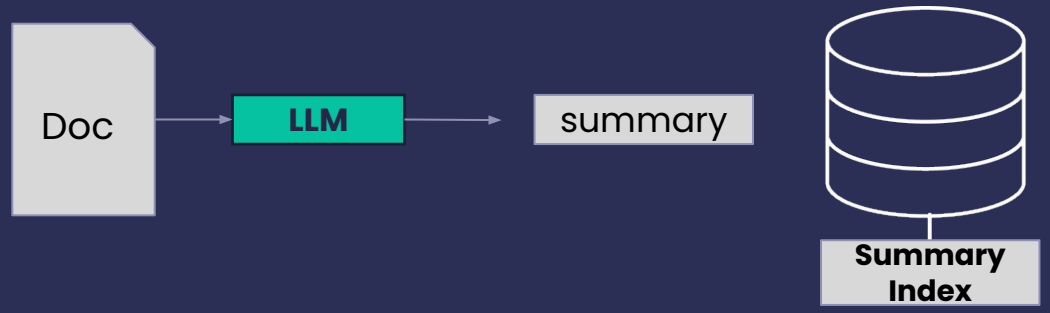
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query
- Each of the *n* documents is embedded into a vector

Hypothetical Document Embeddings – HyDE



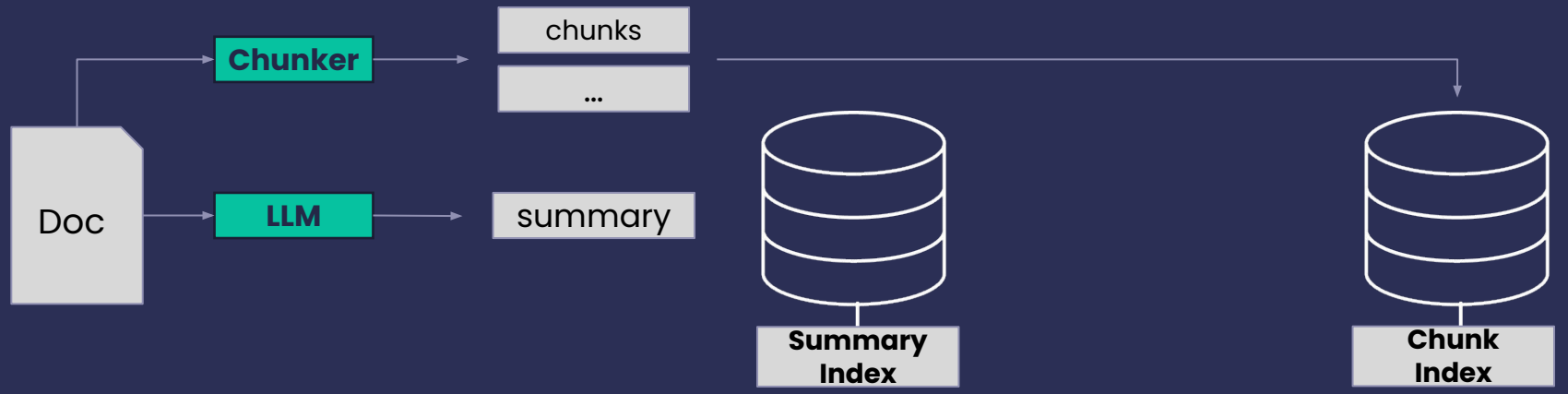
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query
- Each of the *n* documents is embedded into a vector
- You perform an average pooling generating a new query embedding used to search for similar documents instead of the original query

Document Summary Indexing



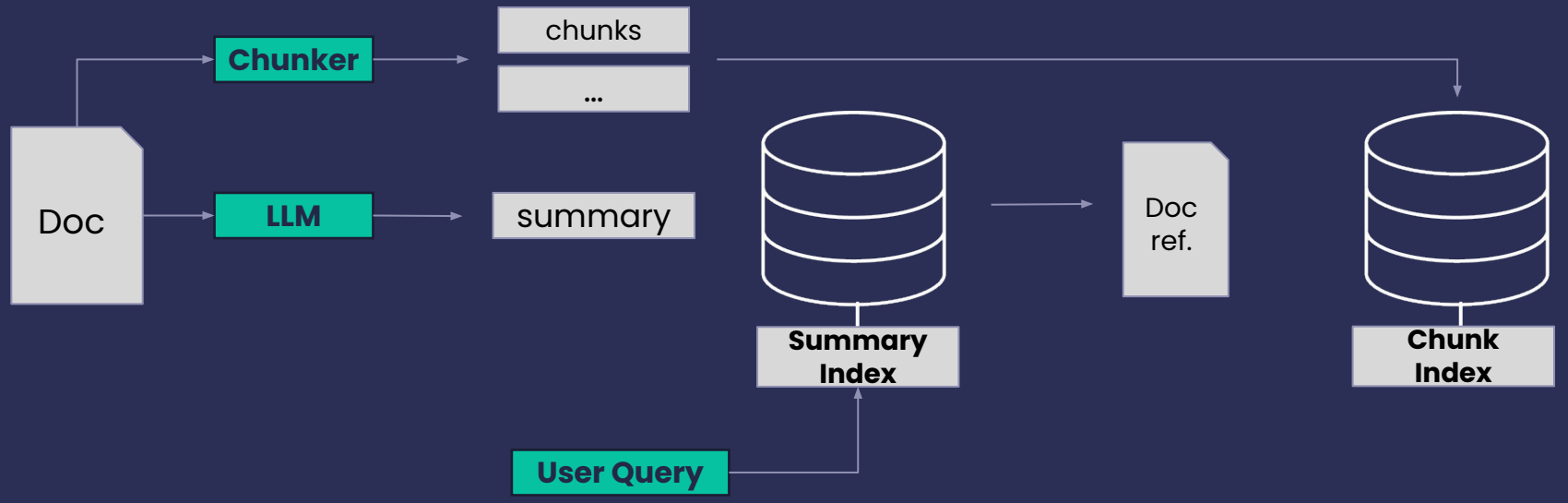
- **Summary Index:** generate a summary for each document with an LLM

Document Summary Indexing



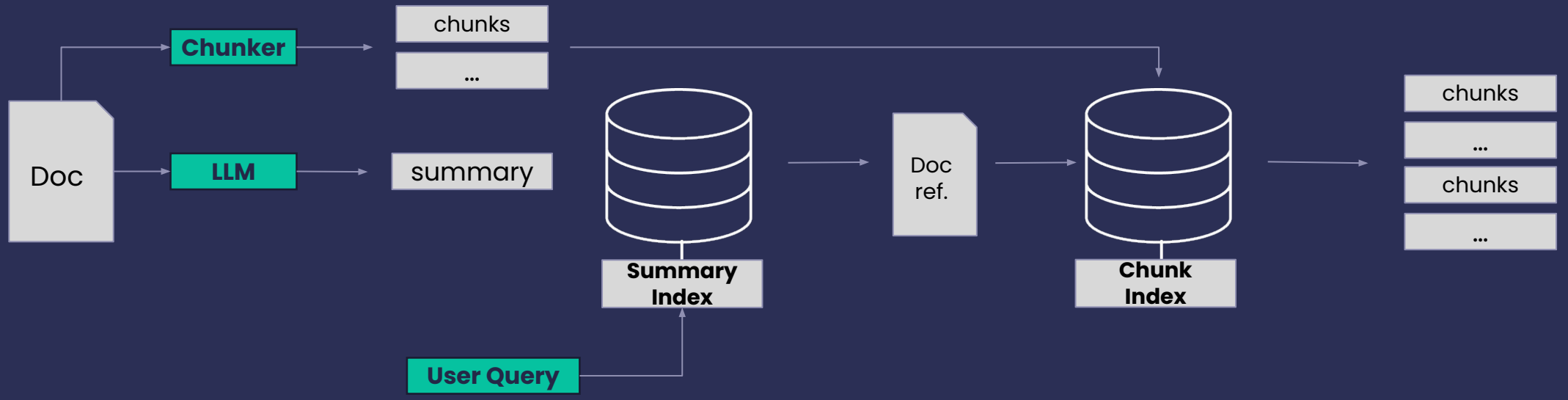
- **Summary Index:** generate a summary for each document with an LLM
- **Chunk Index:** split each document up into chunks

Document Summary Indexing



- **Summary Index:** generate a summary for each document with an LLM
- **Chunk Index:** split each document up into chunks
- Use the **Summary Index** to retrieve top-k relevant documents to the query

Document Summary Indexing



- **Summary Index:** generate a summary for each document with an LLM
- **Chunk Index:** split each document up into chunks
- Use the **Summary Index** to retrieve top-k relevant documents to the query
- Using the document(s) reference retrieve the most relevant chunks

Summary

	Custom Index Structure	ReRanking	Query Rewriting	Combining multiple sources	Relies on a LLM
Sentence-Window Retrieval	X				
Auto-Merging Retrieval	X				
Maximum Margin Relevance		X			
Hybrid Retrieval				X	
Multi-Query			X		X
Hypothetical Document Embeddings			X		X
Document Summary Indexing	X			X	X

Comparative Experiment

- "ARAGOG: Advanced RAG Output Grading" M Eibich, S Nagpal, A Fred-Ojala arXiv preprint, 2024

- **Dataset:**
  - ArXiv preprints covering topics around Transformers and LLMs
  - 13 PDF papers (https://huggingface.co/datasets/jamescalam/ai-arxiv)
  - 107 questions and answers generated with the assistance of an LLM
  - All questions and answers were manually validated and corrected

- **Experiment:**
  - Run the questions over each retrieval technique
  - Compare ground-truth answer with generated answer
  - Semantic Answer Similarity: cos sim embeddings of both answers

Comparative Experiment: ARAGOG

	Semantic Answer Similarity	Specific Parameters
Sentence-Window Retrieval	0.688	window=3
Auto-Merging Retrieval	0.619	threshold=0.5, block_sizes={10, 5}
Maximum Margin Relevance	0.607	lambda_threshold=0.5
Hybrid Retrieval	0.701	join_mode="concatenate"
Multi-Query	0.692	n_variations=3
Hypothetical Document Embeddings	0.642	nr_completions=3
Document Summary Indexing	0.731	-

- sentence-transformers/all-MiniLM-L6-v2
- chunk_size = 15
- split_by = "sentence"
- top_k = 3

LLM

OpenAI: gpt-4o-mini

Takeaways

- Build a dataset for our use case - **50~100 annotated questions**
- Start with the simple RAG approach and set it as your baseline
- Start by exploring “cheap” and simple techniques
- **Sentence-Window Retriever** and **Hybrid Retrieval -** good results and no need for complexing indexing or an LLM
- If none of these produces satisfying results then, explore indexing/retrieval methods based on LLMs

Haystack Implementations

Sentence-Window Retrieval	haystack.components.retrievers.SentenceWindowRetriever
Auto-Merging Retrieval	haystack.components.retrievers.AutoMergingRetriever haystack.components.preprocessors.HierarchicalDocumentSplitter
Maximum Margin Relevance	haystack.components.rankers.SentenceTransformersDiversityRanker
Hybrid Retrieval w/ ReRanking	haystack.components.retrievers.InMemoryEmbeddingRetriever haystack.components.retrievers.InMemoryBM25Retriever haystack.components.joiners.DocumentJoiner (ranking techniques)
Multi-Query	https://github.com/davidsbatista/haystack-retrieval https://haystack.deepset.ai/blog/query-expansion https://haystack.deepset.ai/blog/query-decomposition
Hypothetical Document Embeddings	https://haystack.deepset.ai/blog/optimizing-retrieval-with-hyde
Document Summary Indexing	https://github.com/davidsbatista/haystack-retrieval

Haystack Implementations - SuperComponents

Sentence-Window Retrieval	haystack.components.retrievers.SentenceWindowRetriever
Auto-Merging Retrieval	haystack.components.retrievers.AutoMergingRetriever haystack.components.preprocessors.HierarchicalDocumentSplitter
Maximum Margin Relevance	haystack.components.rankers.SentenceTransformersDiversityRanker
Hybrid Retrieval w/ ReRanking	**Will be soon available as a SuperComponent!**
Multi-Query	**Can be built as a SuperComponent!**
Hypothetical Document Embeddings	**Can be built as a SuperComponent!**
Document Summary Indexing	**Can be built as a SuperComponent!**

SuperComponent in Haystack 2.12.0!

- wrap complex pipelines into reusable (super)components
- easy to reuse them across applications
- Initialize a SuperComponent with a pipeline

References

- "The use of MMR, diversity-based reranking for reordering documents and producing summaries" J Carbonell, J Goldstein - ACM SIGIR 1998
- "ARAGOG: Advanced RAG Output Grading" M Eibich, S Nagpal, A Fred-Ojala arXiv preprint, 2024
- "Advanced RAG: Query Expansion" - Haystack Blog, 2024
- "Advanced RAG: Query Decomposition & Reasoning" - Haystack Blog, 2024
- "Precise Zero-Shot Dense Retrieval without Relevance Labels" Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan- *ACL 2023*
- "A New Document Summary Index for LLM-powered QA Systems", Jerry Liu 2023

Code and experiments

Haystack

Code + Slides

www.davidsbatista.net

Comparative Experiment

- HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, Christopher D. Manning

- **Dataset:**
  - Question-Answering dataset over Wikipedia articles
  - Very short answers, e.g.:
    - *'Who is older Glenn Hughes or Ross Lynch?'*
    - *'Are Gin and tonic and Paloma both cocktails based on tequila?'*
  - Select the first 100 questions from `hotpot_train_v1.1.json`

- **Experiment:**
  - Evaluate the retrievers: Precision, Recall and Fallout
  - Evaluate generated answer: Semantic Answer Similarity

Comparative Experiment: HotpotQA

	Recall	Precision	Fall-out	Semantic Answer Similarity	Parameters
Sentence-Window Retrieval	0.73	0.49	0.51	0.688	window=3
Auto-Merging Retrieval	0.36	0.26	0.74	0.376	0
Maximum Margin Relevance	0.73	0.48	0.52	0.500	lambda_threshold=0.75
Hybrid Retrieval	0.81	0.40	0.60	0.512	join_mode="concatenate"
Multi-Query	0.75	0.36	0.64	0.515	n_variations=3
Hypothetical Document Embeddings	0.77	0.53	0.47	0.529	nr_completions=3
Document Summary Indexing	0.72	0.47	0.53	0.514	0

- sentence-transformers/all-MiniLM-L6-v2
- chunk_size = 15
- split_by = "sentence"
- top_k = 3

LLM

OpenAI: gpt-4o-mini

Maximum Marginal Relevance (MMR)

$$\text{MMR} = \arg\max_{d_i \in D \setminus R} [\lambda \cdot \text{Sim}_1(d_i, q) - (1 - \lambda) \cdot \max_{d_j \in R} \text{Sim}_2(d_i, d_j)]$$

- D is the set of all candidate documents
- R is the set of already selected documents
- q is the query
- Sim_1 is the similarity function between a document and the query
- Sim_2 is the similarity function between two documents
- d_i and d_j are documents in D and R respectively
- λ is a parameter that controls the trade-off between relevance and diversity

Number of LLM calls

- **Multi-Query:**
  - Each query results in 2 LLM calls
- **Hypothetical Document Embeddings**
  - Each query results in 2 LLM calls
- **Document Summary Indexing**
  - Dependent on the detail parameter
  - LLM_calls = 1 + detail * (X / minimum_chunk_size - 1)