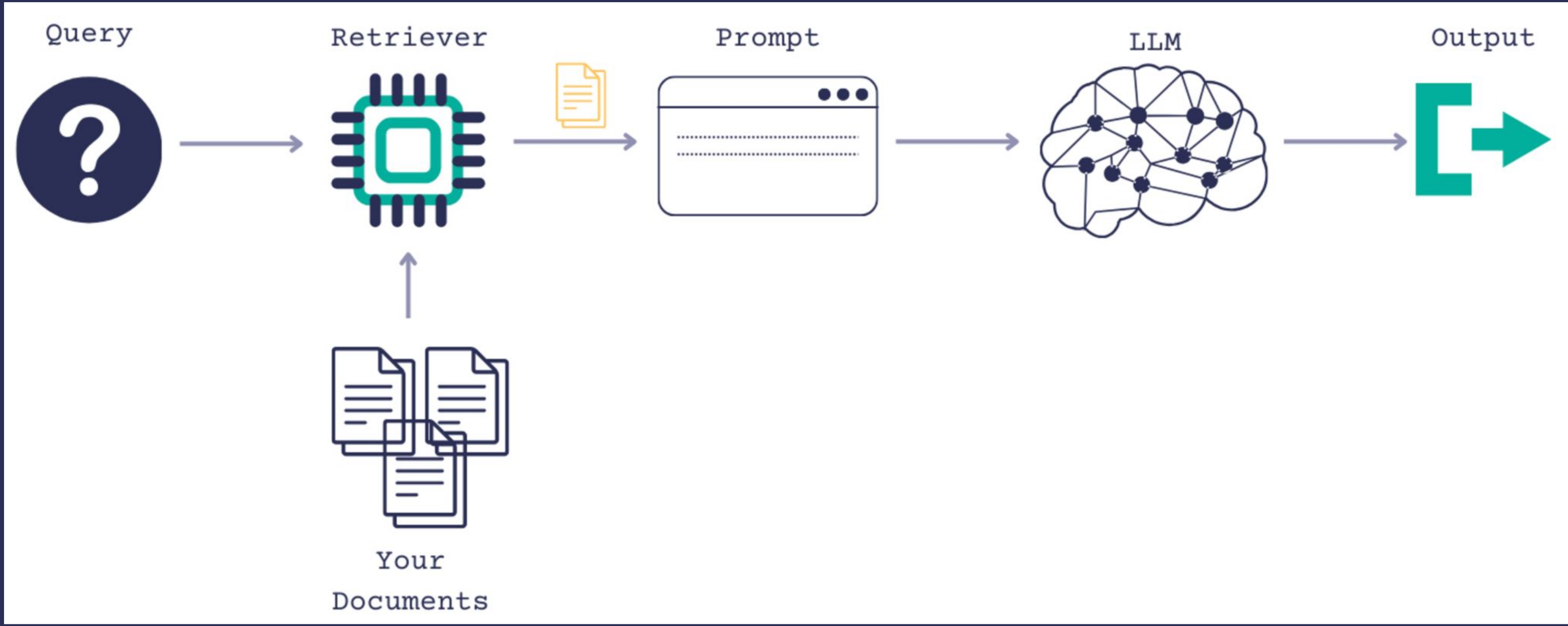
haystack
by deepset

Retrieving with Haystack

by David S. Batista

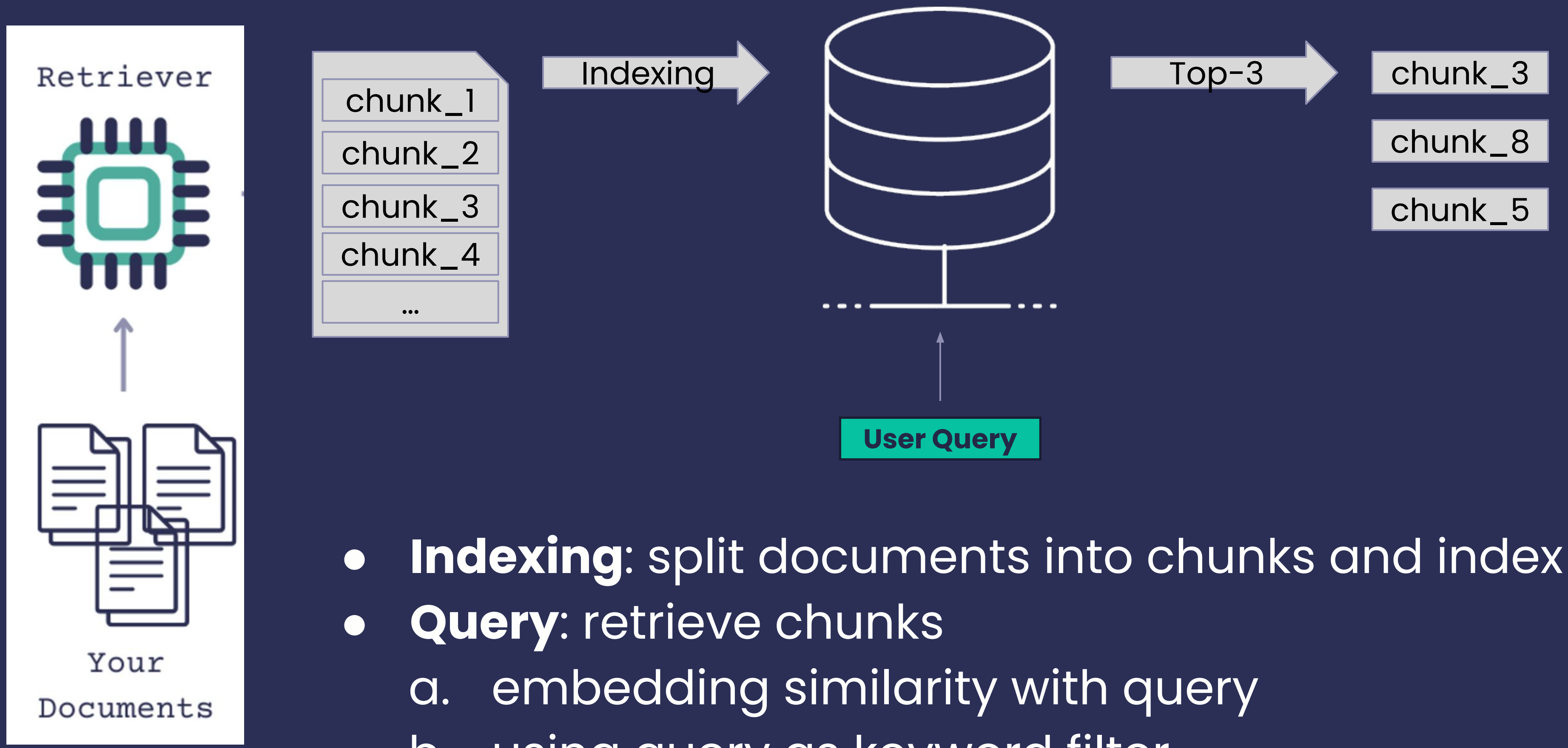
November 2024

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

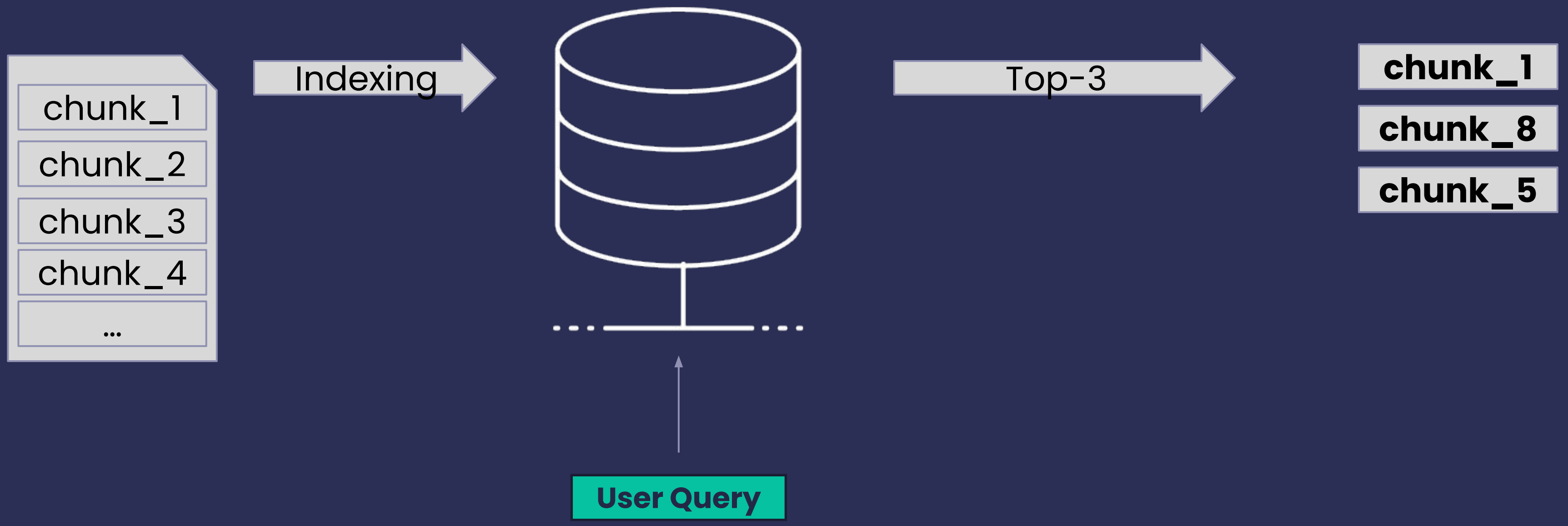
Outline

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

Classical Techniques

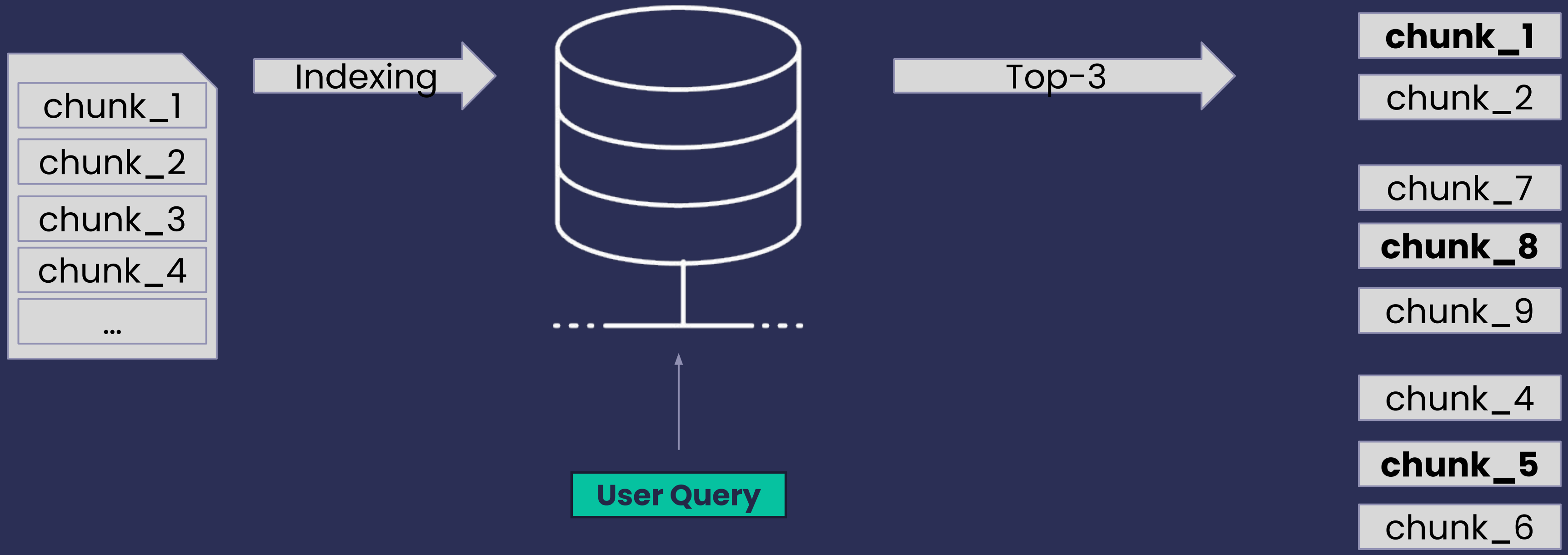
- Sentence-Window Retrieval
- Auto-Merging Retrieval
- Maximum Marginal Relevance
- Hybrid Retrieval (with/ Reciprocal Rank fusion)

Sentence-Window Retrieval



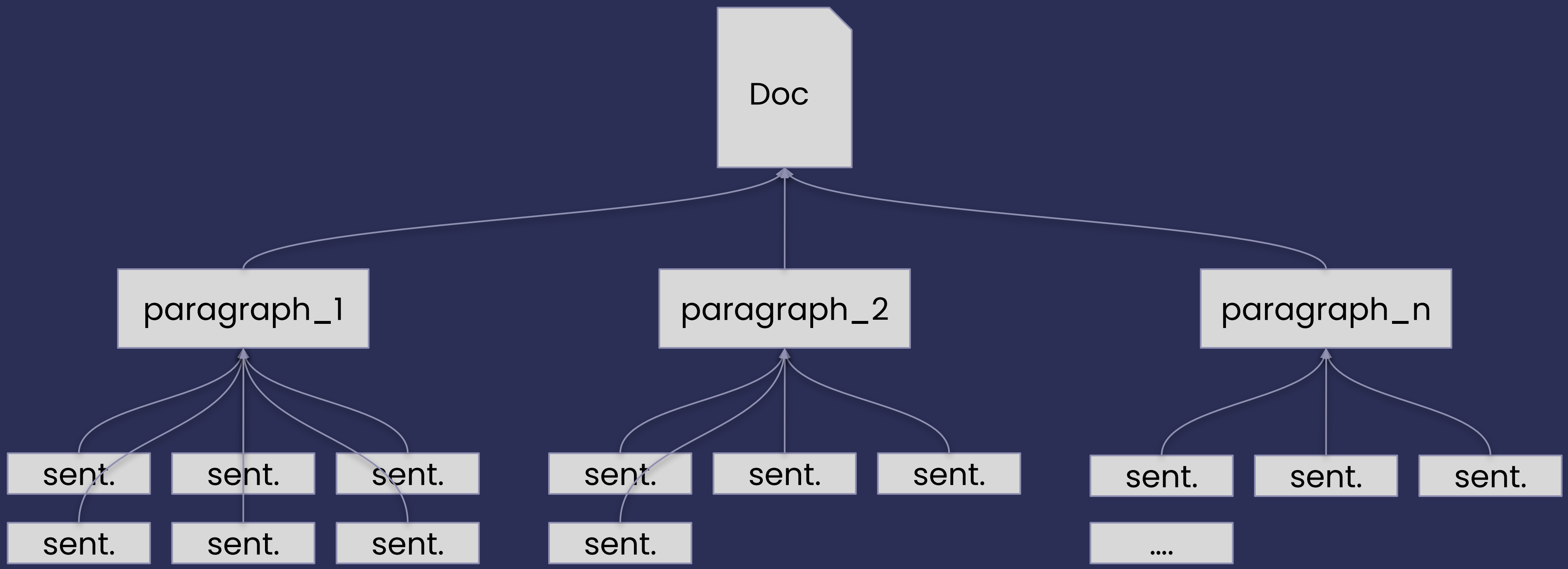
- Retrieve the chunks before and after the matching chunk

Sentence-Window Retrieval



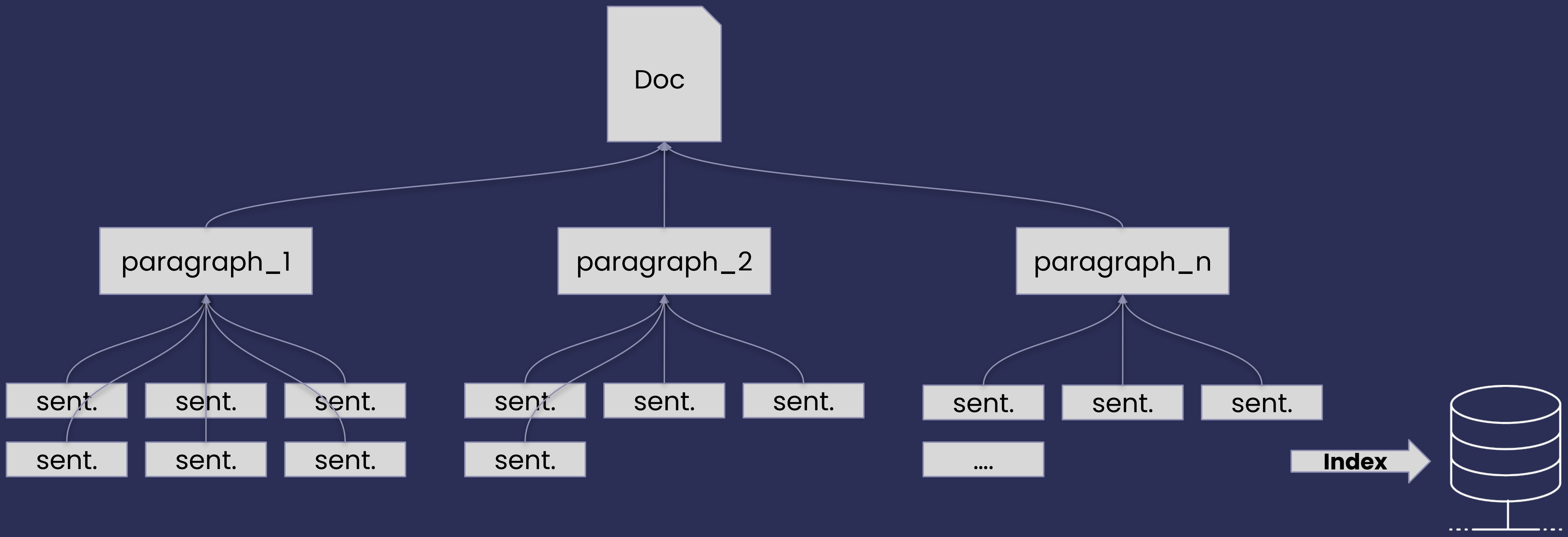
- Retrieve the chunks before and after the matching chunk
- A simple way to gather more context
- Indexing needs to preserve the order of the chunks

Auto-Merging Retrieval



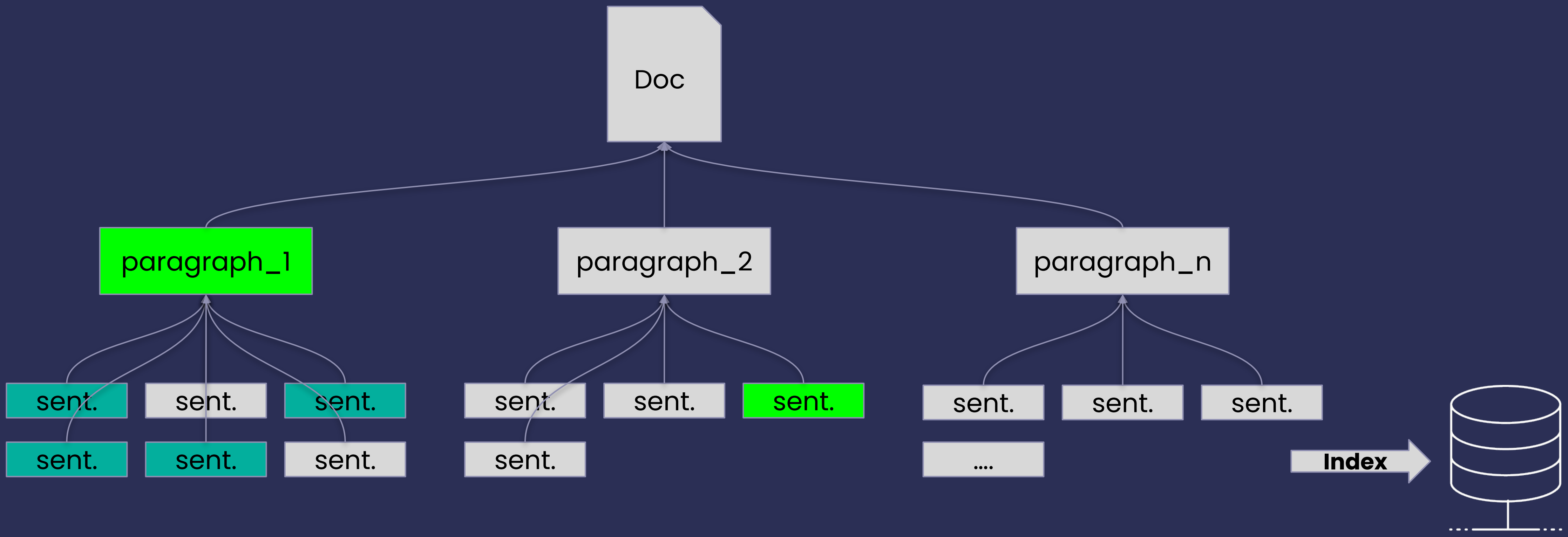
- Transform documents into an Hierarchical Tree structure

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

- But what in cases where there is high number of potentially relevant documents, highly redundant with each other or containing partially or fully duplicative information?

- 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)

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

Each retrieved document is scored:

- Similarity between a candidate document and the query

Maximum Marginal Relevance (MMR)

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

Each retrieved document is scored:

- 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.

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 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

Maximum Marginal Relevance (MMR)

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

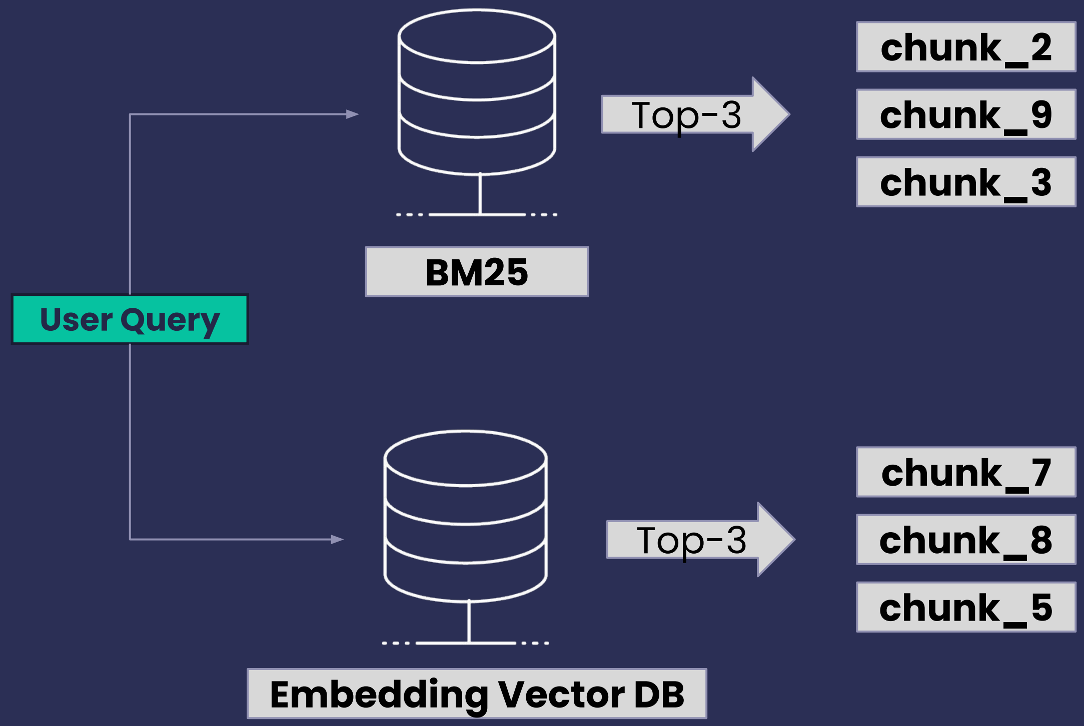
Each retrieved document is scored:

- 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

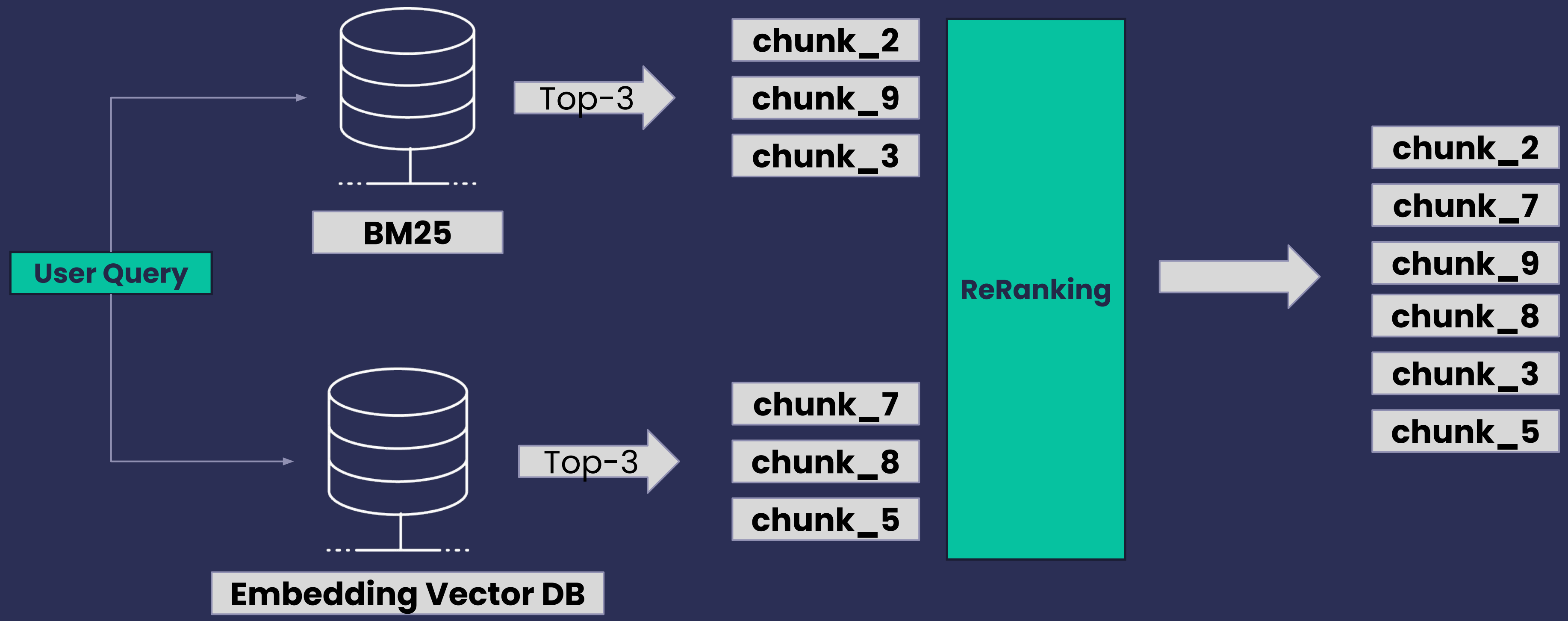
- Combines multiple search techniques
- keyword-based (BM25)

Hybrid Retrieval + Reranking



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

Hybrid Retrieval + Reranking

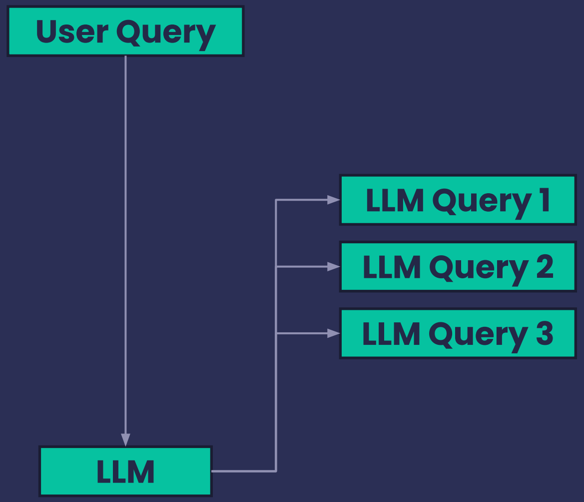


- Combines multiple search techniques
- keyword-based (BM25) and semantic-based (embedding vector)
- Rank-merge results

LLM-based Techniques

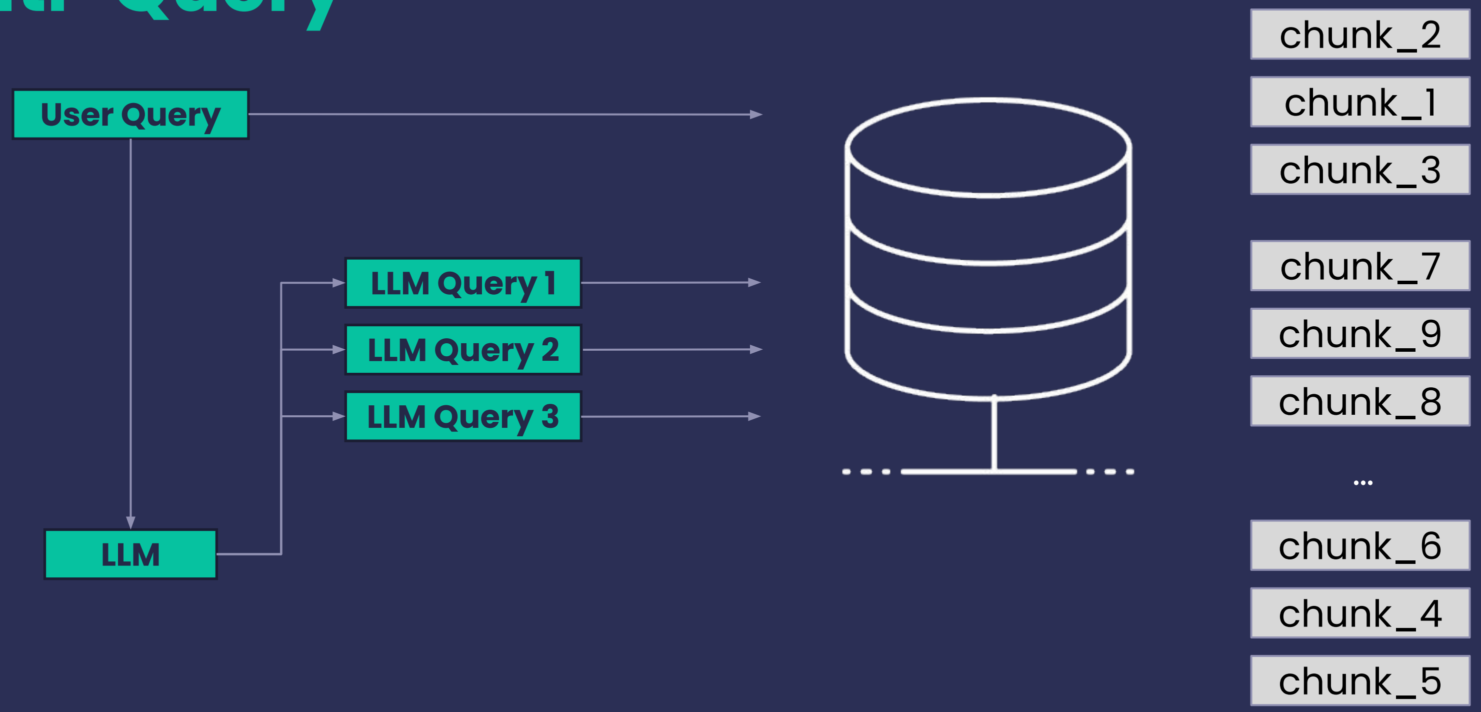
- Multi-Query
- Hypothetical Document Embeddings - HyDE
- Document Summary Indexing

Multi-Query



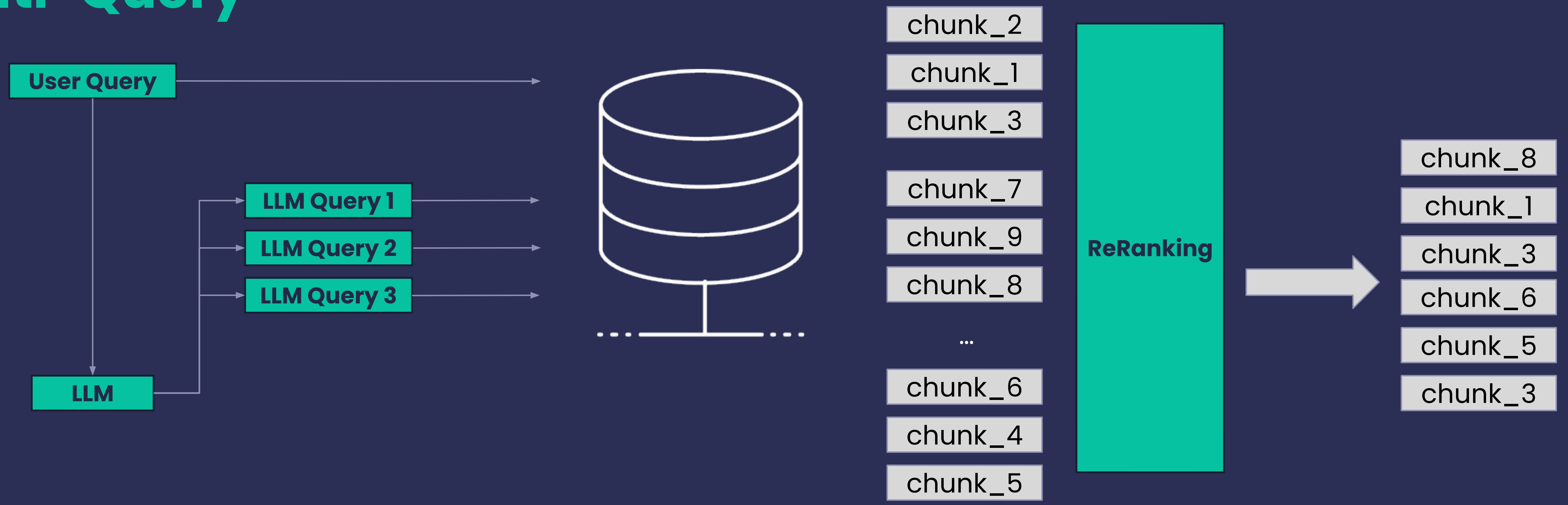
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions

Multi-Query



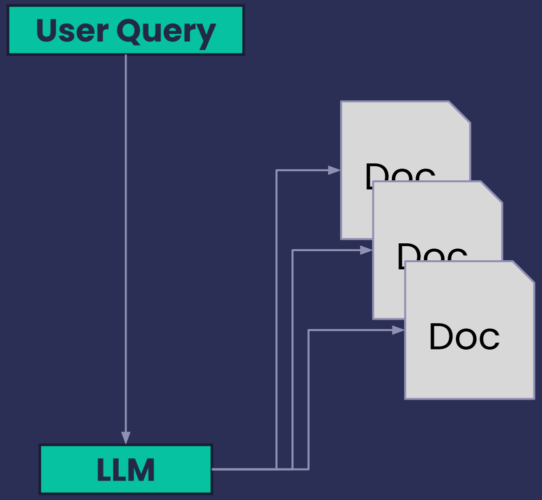
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions
- Each new query is used for an individual retrieval processes

Multi-Query



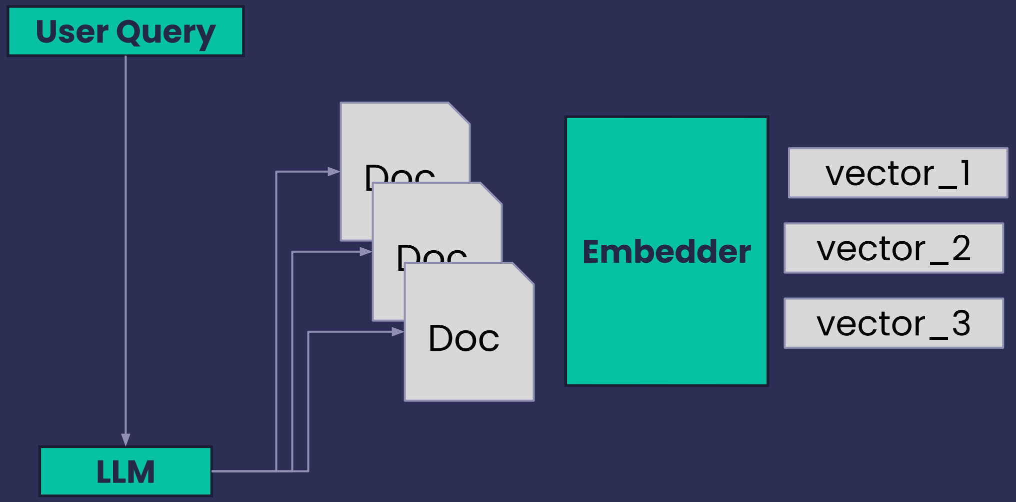
- Expand a user query into *n* similar queries reflecting the original intent
- ..or break-down a complex query into individual questions
- Each new query is used for an individual retrieval processes
- Re-ranking process over all retrieved chunks

Hypothetical Document Embeddings - HyDE



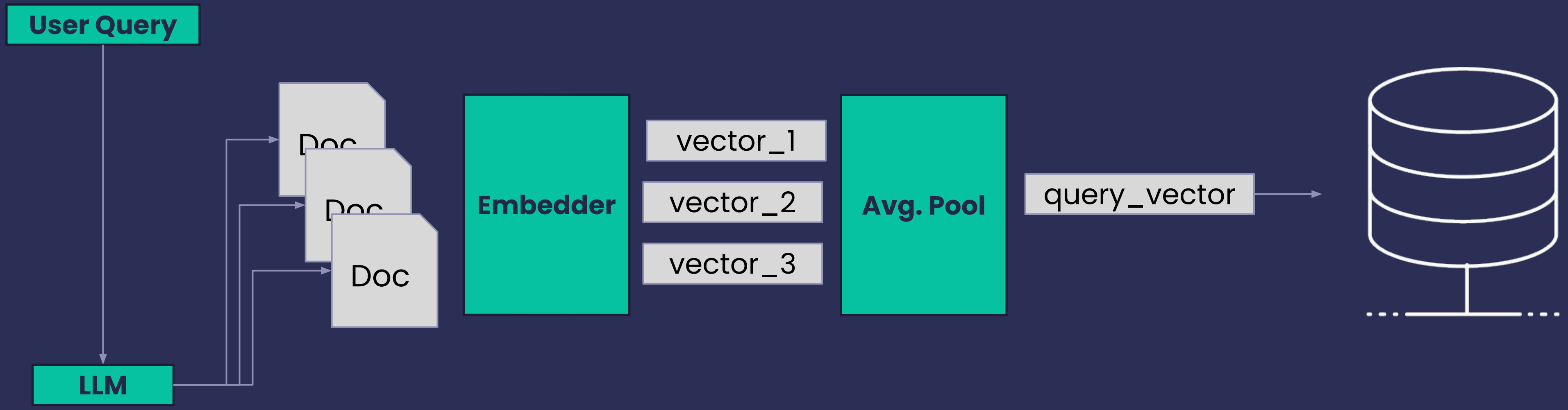
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query

Hypothetical Document Embeddings - HyDE



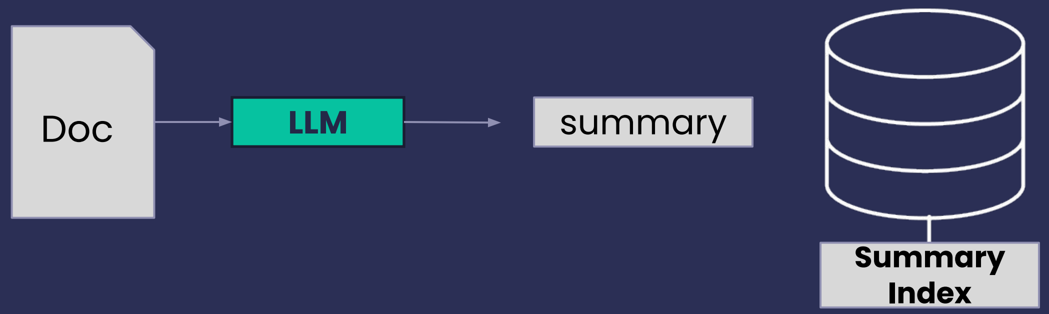
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query
- Each of the *n* documents is embedded into a vector

Hypothetical Document Embeddings - HyDE



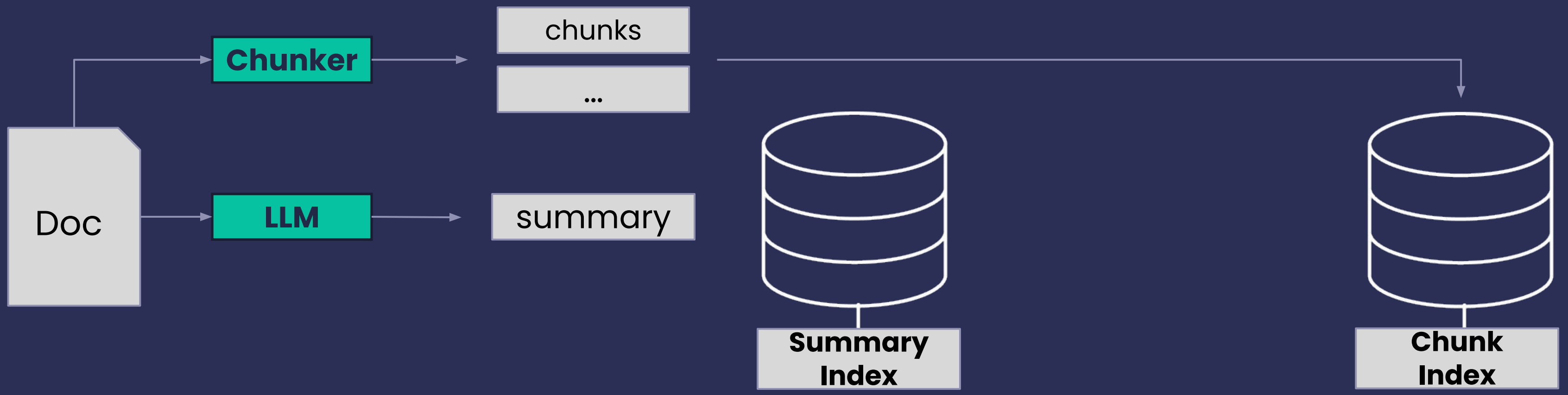
- Given a user query, use a LLM to generate *n* "hypothetical" (short) documents whose content would ideally answer the query
- Each of the *n* documents is embedded into a vector
- You perform an average pooling generating a new query embedding used to search for similar documents instead of the original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

Document Summary Indexing



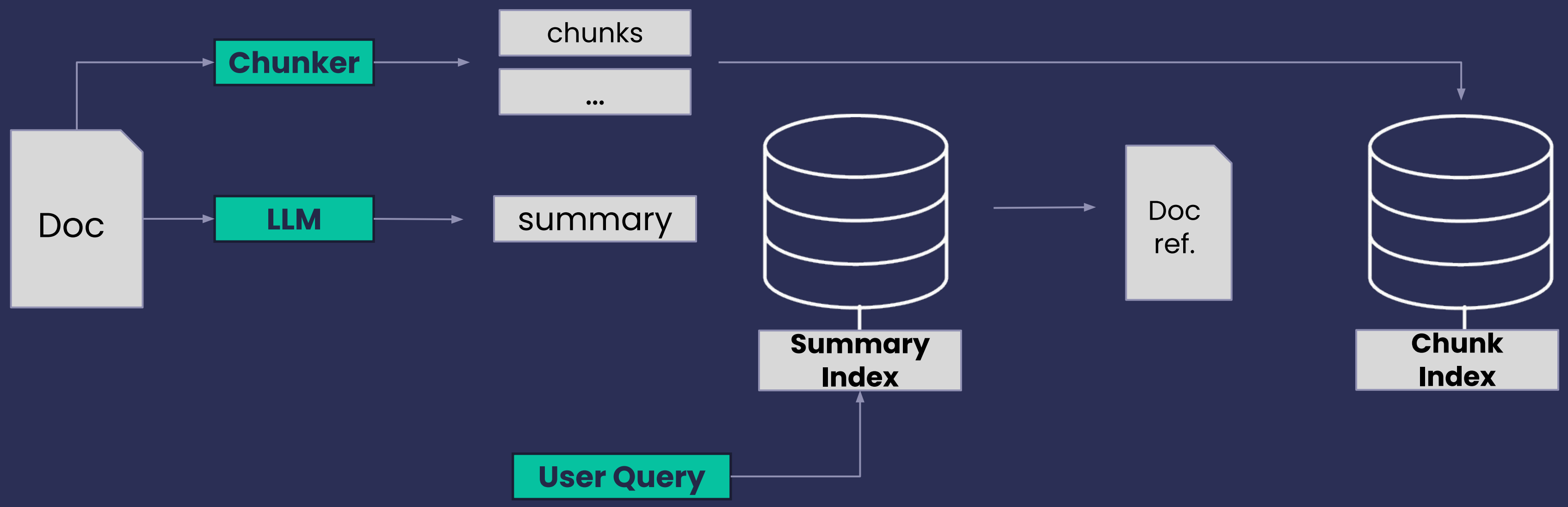
- **Summary Index:** generate a summary for each document with an LLM

Document Summary Indexing



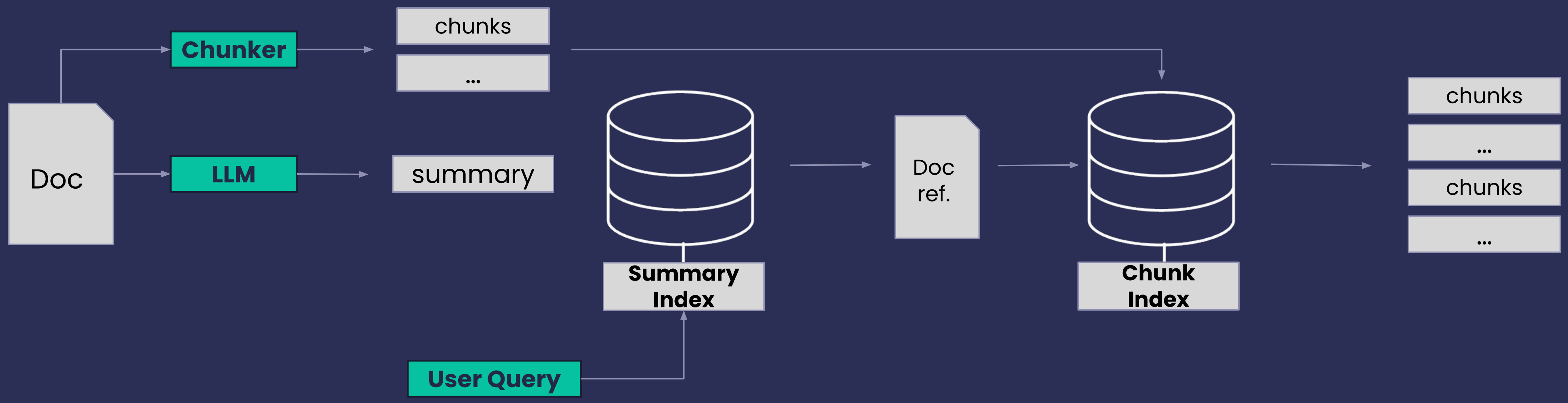
- **Summary Index:** generate a summary for each document with an LLM
- **Chunk Index:** split each document up into chunks

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 GPT-4
  - All validated and corrected by humans

- **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

	Semantic Answer Similarity	Specific Parameters
Sentence-Window Retrieval	0.700	window=3
Auto-Merging Retrieval	0.505	threshold=0.5, block_sizes={10, 5}
Maximum Margin Relevance	0.670	lambda_threshold=0.5
Hybrid Retrieval	0.699	join_mode="concatenate"
Multi-Query	0.620	n_variations=3
Hypothetical Document Embeddings	0.693	nr_completions=3
Document Summary Indexing	0.731	-

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

LLM

gpt-4o-mini (OpenAI)

https://github.com/davidsbatista/haystack-retrieval

Haystack Implementations

Sentence-Window Retrieval	haystack.components.retrievers.SentenceWindowRetriever
Auto-Merging Retrieval	haystack_experimental.components.retrievers.AutoMergingRetriever haystack_experimental.components.splitters.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

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

Haystack 2.x

- Python open-source framework to build LLM-based applications
- Production focused

https://github.com/deepset-ai/haystack

Implementation and Experiments

https://github.com/davidsbatista/haystack-retrieval

Maximum Marginal Relevance (MMR)

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

- 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