



Five Techniques for Improving RAG Chatbots

Improving Retrievers to Build High-Quality Conversational Assistants

Nikita Kozodoi, PhD

13.02.2024



About Me



<https://kozodoi.me>

- **Applied Scientist** at Amazon Web Services
- Building **Generative AI** solutions across industries
- Working on the frontier of **research & business**
- Earned **PhD in ML** for Credit Risk Analytics
- Won 18 **Kaggle competition medals**



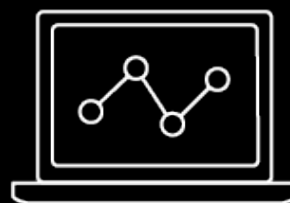
My Team: Generative AI Innovation Center



Design

Design guidance:

- Select the GenAI use case with the **highest business impact**
- Design how to develop, train, and **deploy it to production**



Deploy

Deploy recommended solutions:

- Develop and fine-tune a GenAI solution to **meet your business objectives** and demonstrate what's possible



Drive

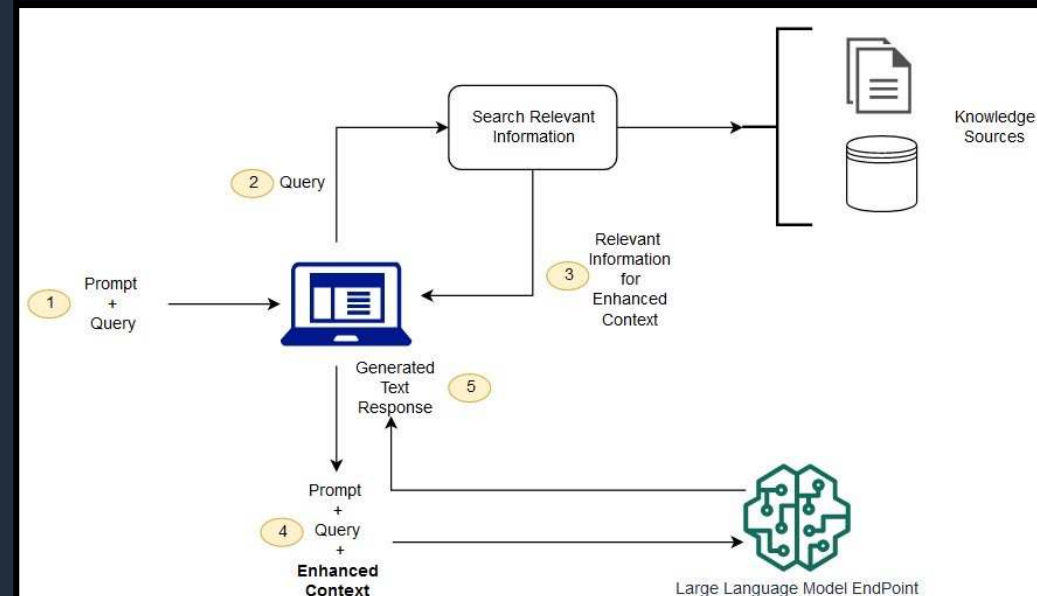
Drive adoption:

- **Accelerate stickiness and adoption** with a path to production for your GenAI solution integrated into your application.

<https://aws.amazon.com/generative-ai/innovation-center/>

Agenda

- Intro to RAG Chatbots
- Techniques to Improve Retrieval
 - #1: Small-to-Big Retrieval
 - #2: Leveraging Meta-Data
 - #3: Hybrid Search
 - #4: Query Rewriting & Expansion
 - #5: Document Reranking



Intro to RAG Chatbots

(RAG = Retrieval Augmented Generation)



Search vs Q&A Assistant

What is self-attention?

Send

[1] Attention is all you need

Attention mechanisms have become an integral part of compelling sequence modeling...

[2] Mistral 7B

The number of operations in vanilla attention is quadratic in the sequence length...

Search



What is self-attention?

Send

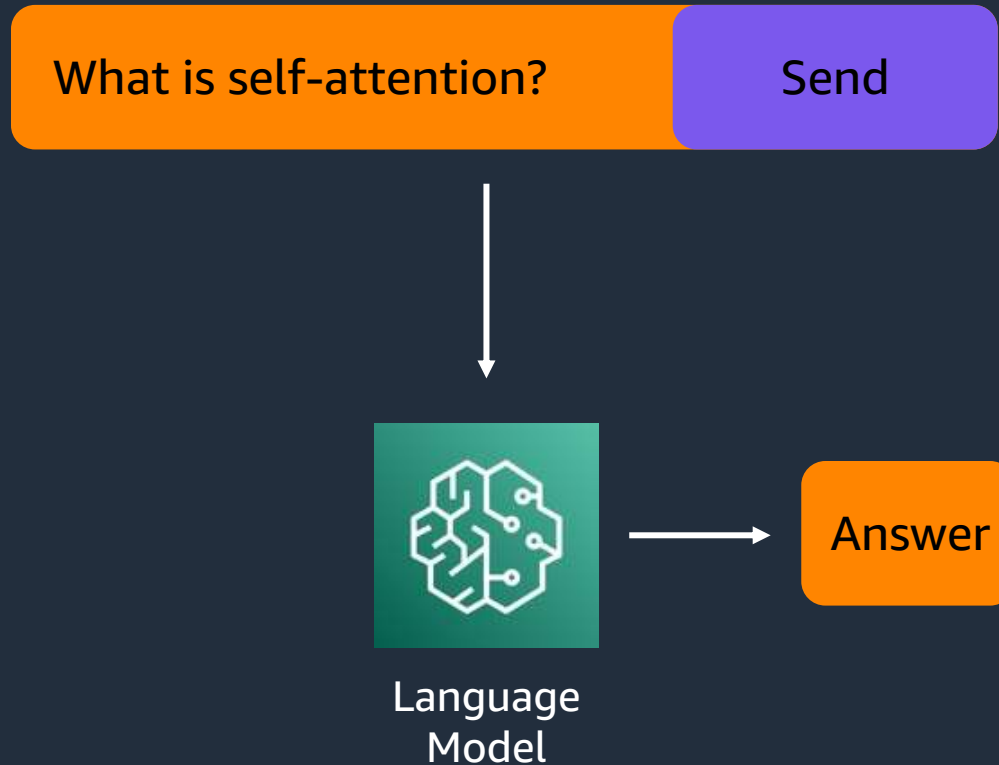
Self-attention is a mechanism used in DL models to capture dependencies and relationships within input sequences [1].

[1] Attention is all you need

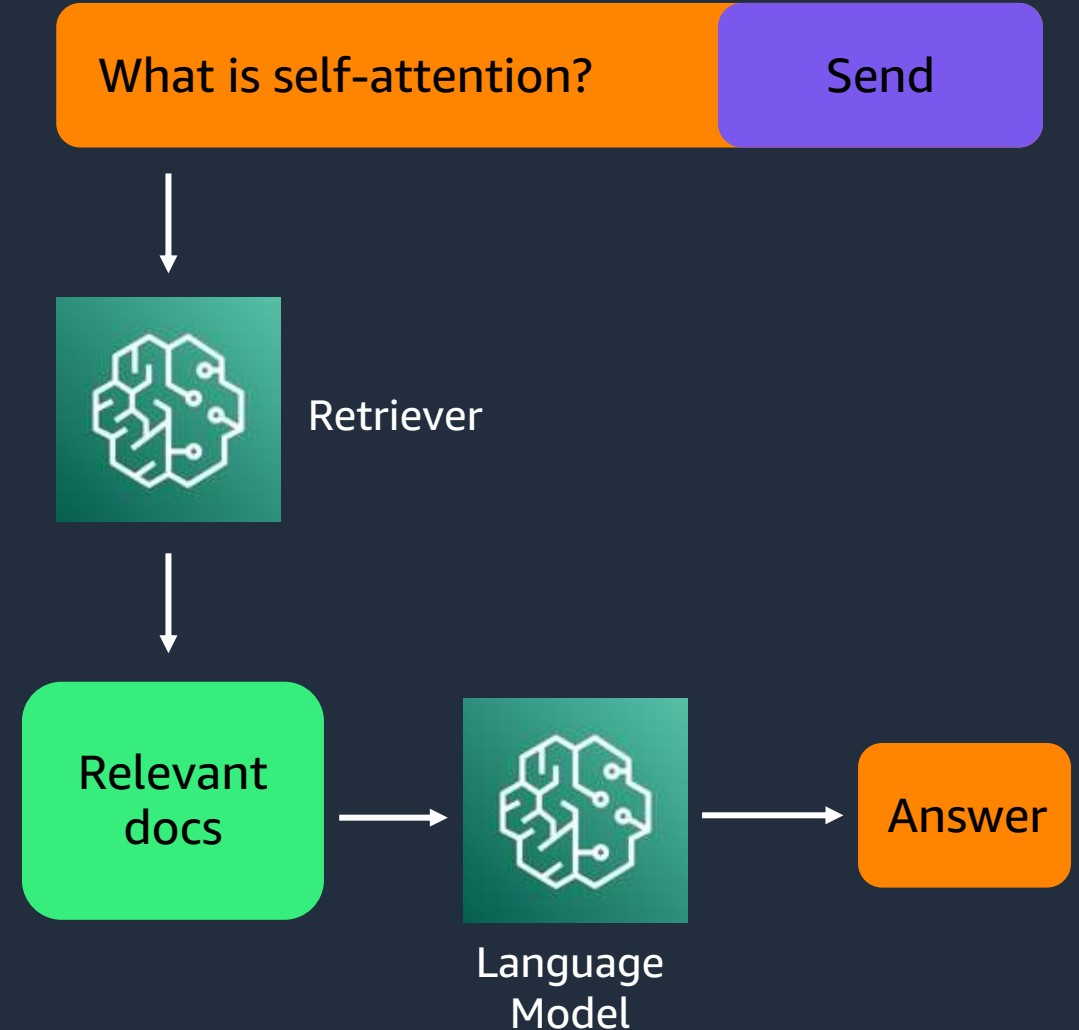
Attention mechanisms have become an integral part of compelling sequence modeling...

Q&A Assistant

Assistant with/without RAG

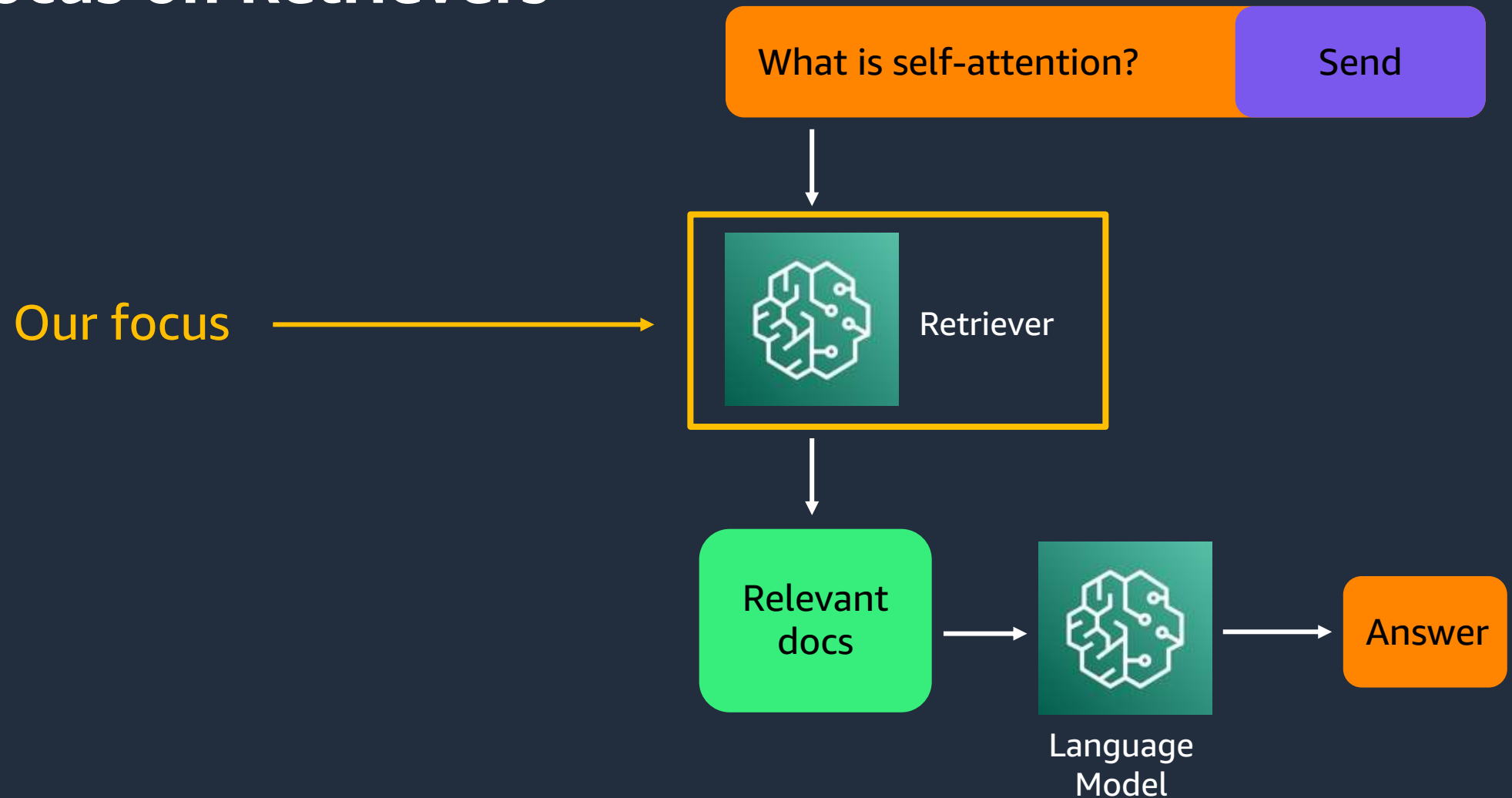


Assistant with no RAG



Assistant with RAG

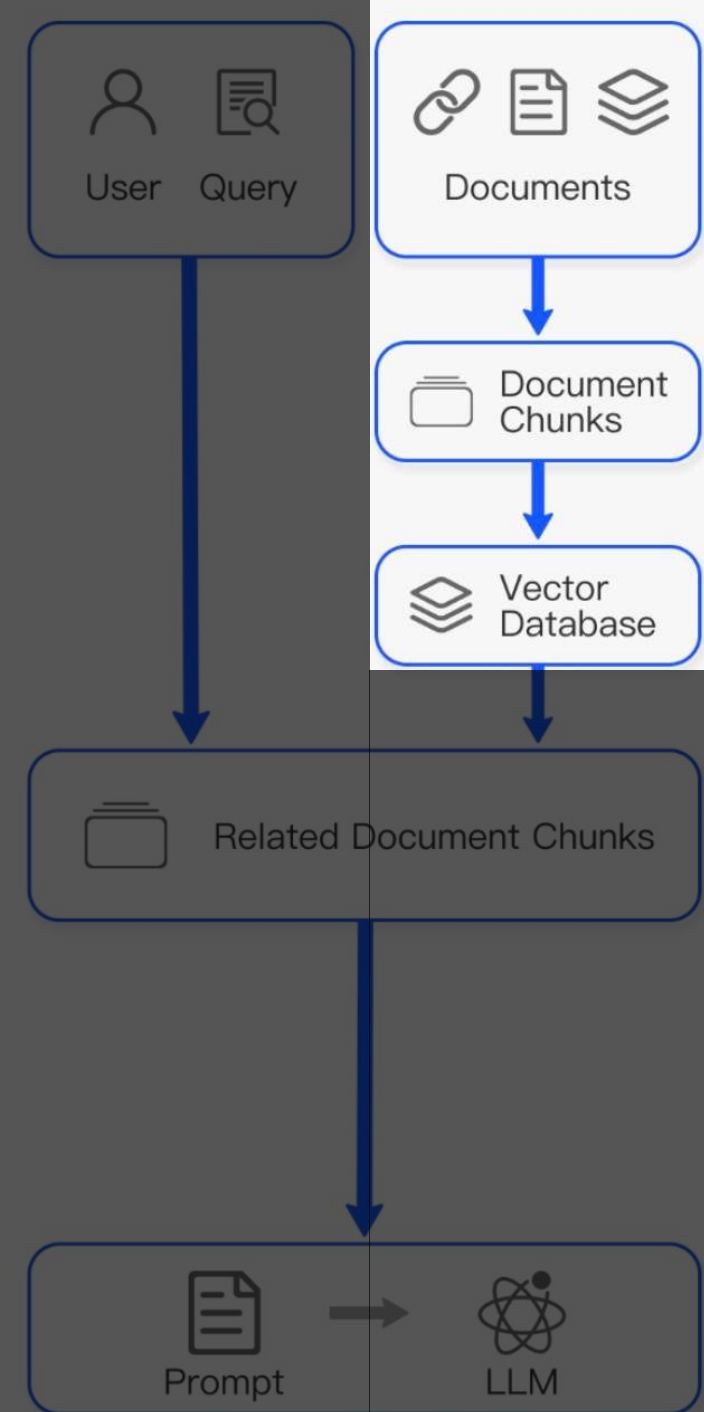
Let's Focus on Retrievers



RAG Chatbot Architecture

1. Indexing stage:

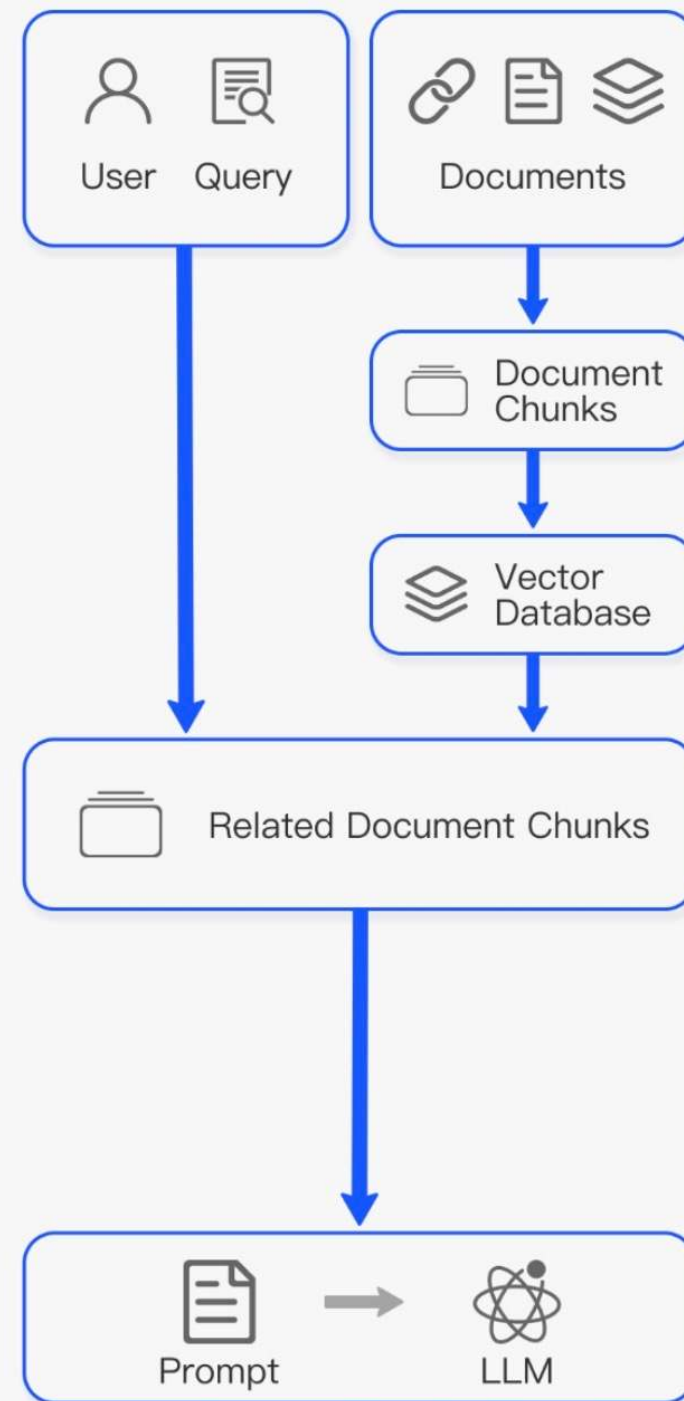
- Each document is parsed into text
- Texts are split into chunks (e.g. paragraphs)
- Each chunk is embedded into a vector
- Vectors and text chunks are stored in a database



RAG Chatbot Architecture

2. Q&A stage:

- User question is embedded into a vector
- Top-K closest chunks are fetched from the database
- User question and chunks are shown to LLM
- LLM provides a response to the user



Techniques to Improve Retrievers

#1. Small-to-Big Retrieval

1. Small-to-Big Chunk Retrieval

Idea: use small chunk size on embedding stage and large size on Q&A stage

Why:

- Embedding very large chunks adds **too much noise** => poor recall
- Embedding small chunks leads to **very limited context** => incomplete answers

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 16]. In all but a few cases [22], however, such attention mechanisms are used in conjunction with a recurrent network. In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state...

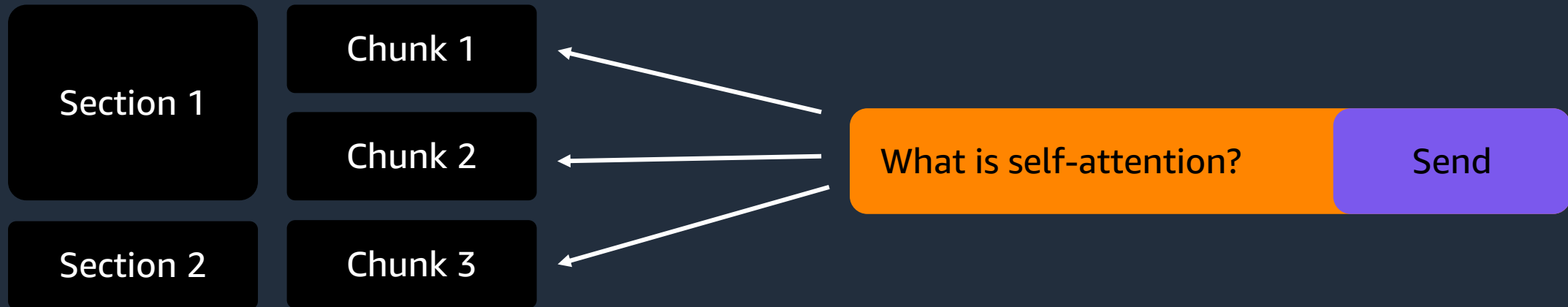
VS

Attention mechanisms have become an integral part of compelling sequence modeling...

1. Small-to-Big Chunk Retrieval

Solution:

- Use a smaller chunk size on the embedding stage
- Append “neighbor” chunks to extend the chunk before feeding to LLM



1. Small-to-Big Chunk Retrieval

Solution:

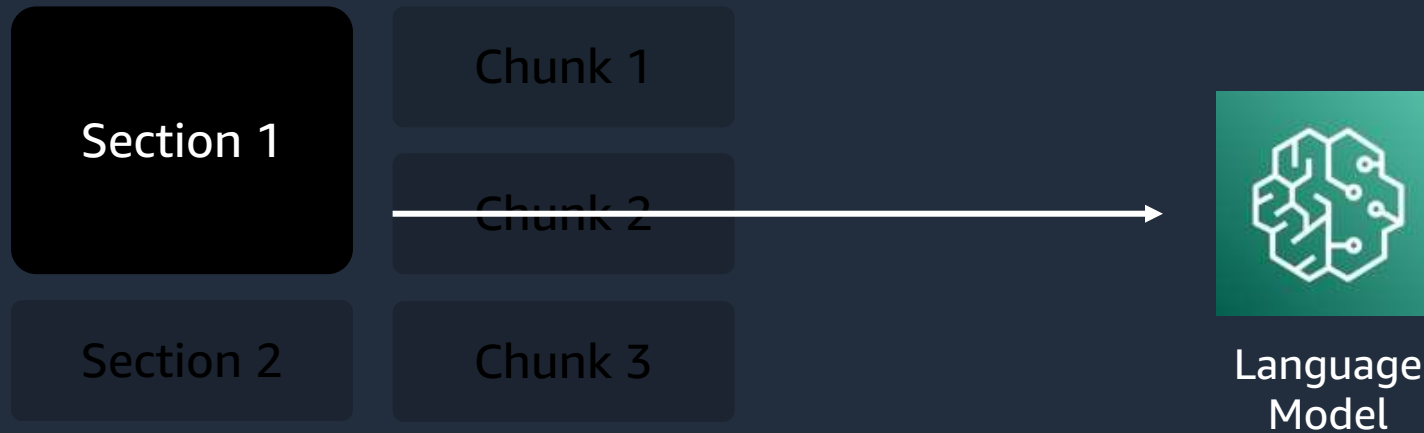
- Use a smaller chunk size on the embedding stage
- Append “neighbor” chunks to extend the chunk before feeding to LLM



1. Small-to-Big Chunk Retrieval

Solution:

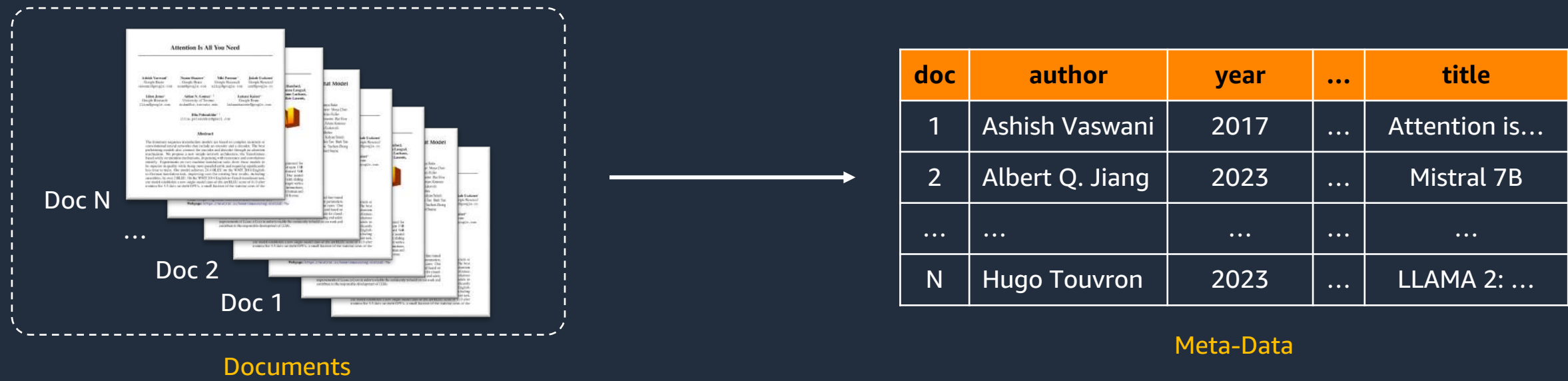
- Use a smaller chunk size on the embedding stage
- Append “neighbor” chunks to extend the chunk before feeding to LLM



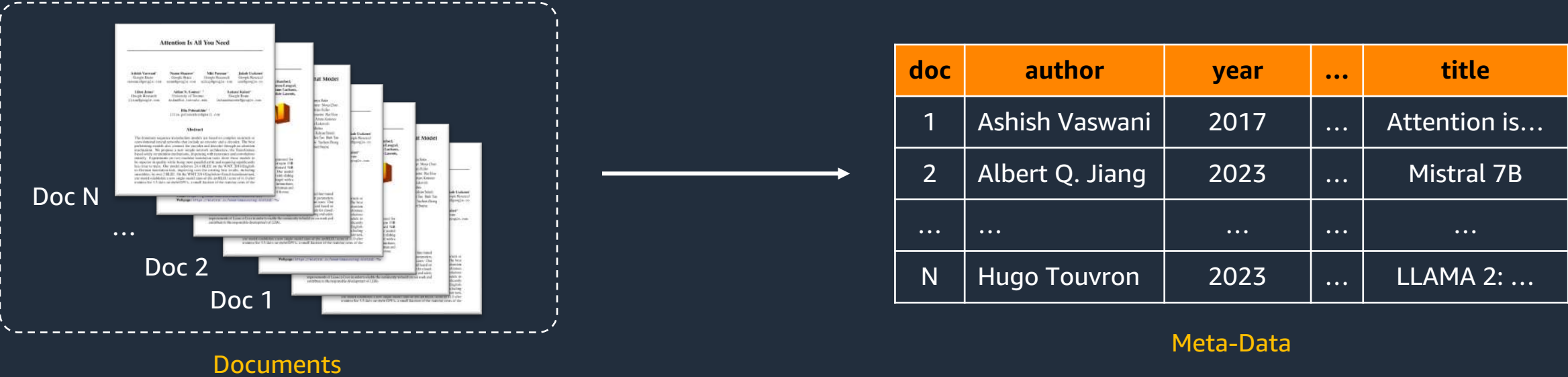
Techniques to Improve Retrievers

#2. Leveraging Meta-Data

2. Leveraging Document Meta-Data



2. Leveraging Document Meta-Data



- Meta-data can start as simple as the file name and path
- You can also use LLM to generate meta-data

2.1. Use Filters Based on Meta-Data

Idea: let user specify filters before asking the question

Why: filters reduce the search space, leading to more precise retrieval

What is self-attention?

Send

Year: 2017

Title: Attention is all you need

2.2. Include Meta-Data in Chunk Text

Idea: append document meta-data as a string to chunks before embedding

Why: relevant meta-data text improves search in the embedding space

Attention mechanisms have
become an integral part of
compelling sequence modeling...

Chunk (Before)



Year: 2017
Title: Attention is all you need

Attention mechanisms have
become an integral part of
compelling sequence modeling...

Chunk (After)

Techniques to Improve Retrievers

#3. Hybrid Search

3. Hybrid Search

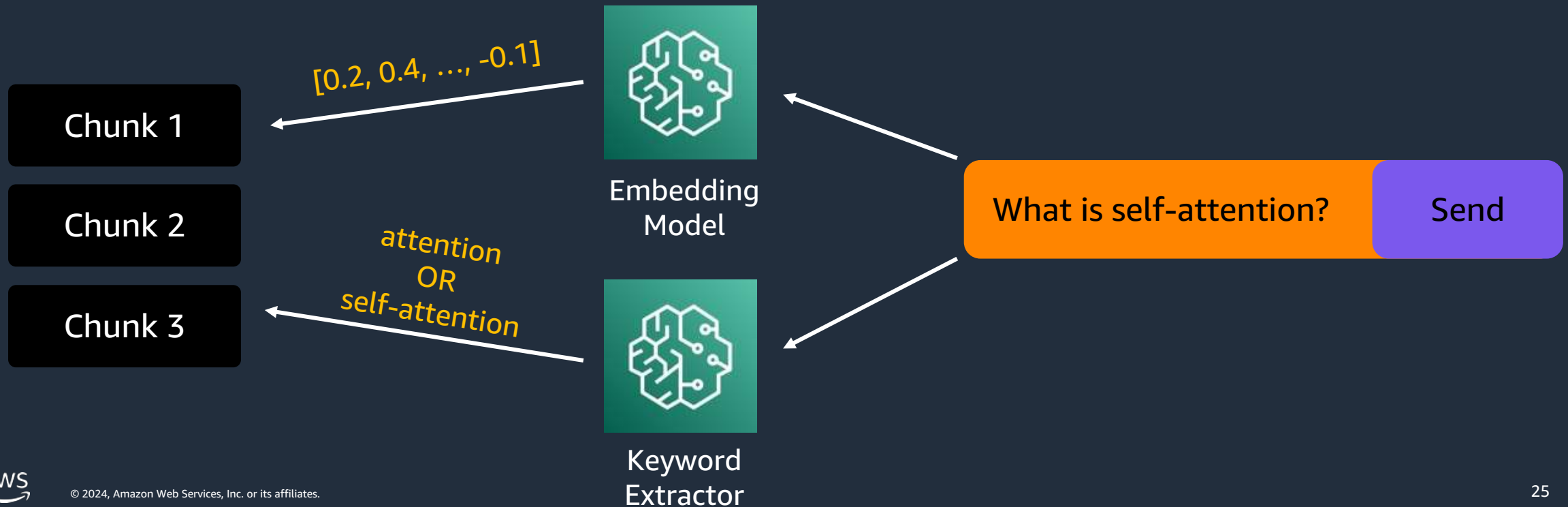
Idea: combine **vector-based** search and **keyword-based** search

Why:

- Consider docs with domain-specific syntax and abbreviations
- Such texts may not be poorly represented in the embedding model data
- Including a few docs from **keyword-based search** helps to address this gap

3. Hybrid Search

Idea: combine **vector-based** search and **keyword-based** search



Techniques to Improve Retrievers

#4. Query Rewriting & Expansion

4. User Query Rewriting

Idea: reformulate user question into a better-structured search query

Why:

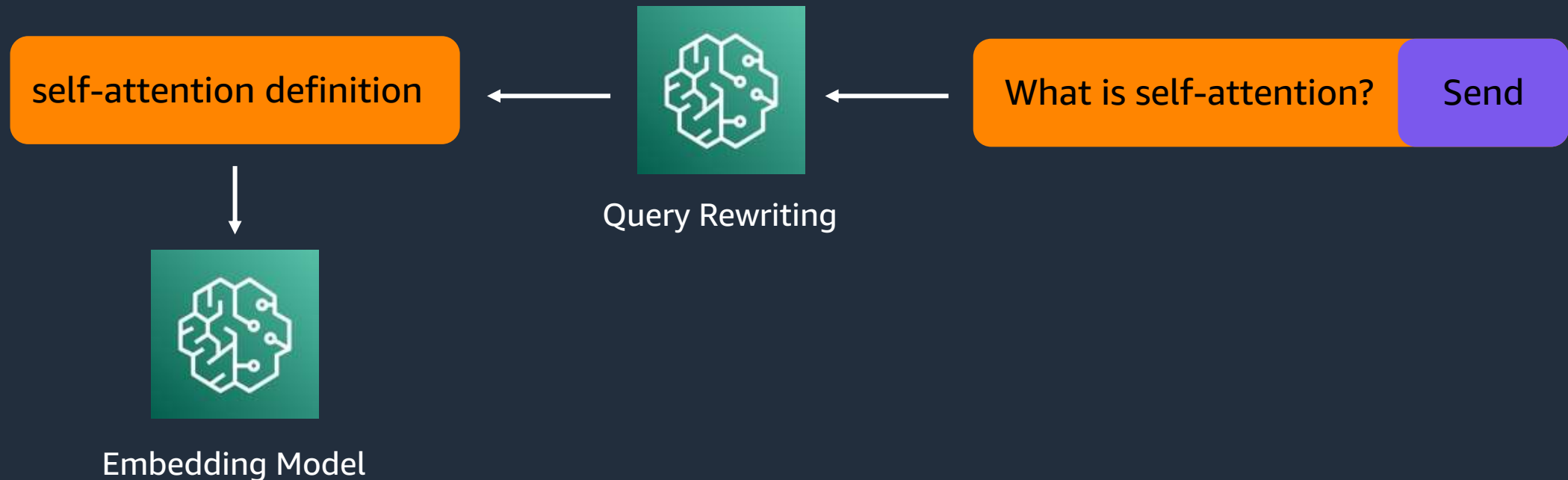
- Users are not always very good at formulating specific questions
- Especially relevant for follow-up conversations that assume certain context

Hey, can you tell me about self-attention please? I am struggling to understand it...

Send

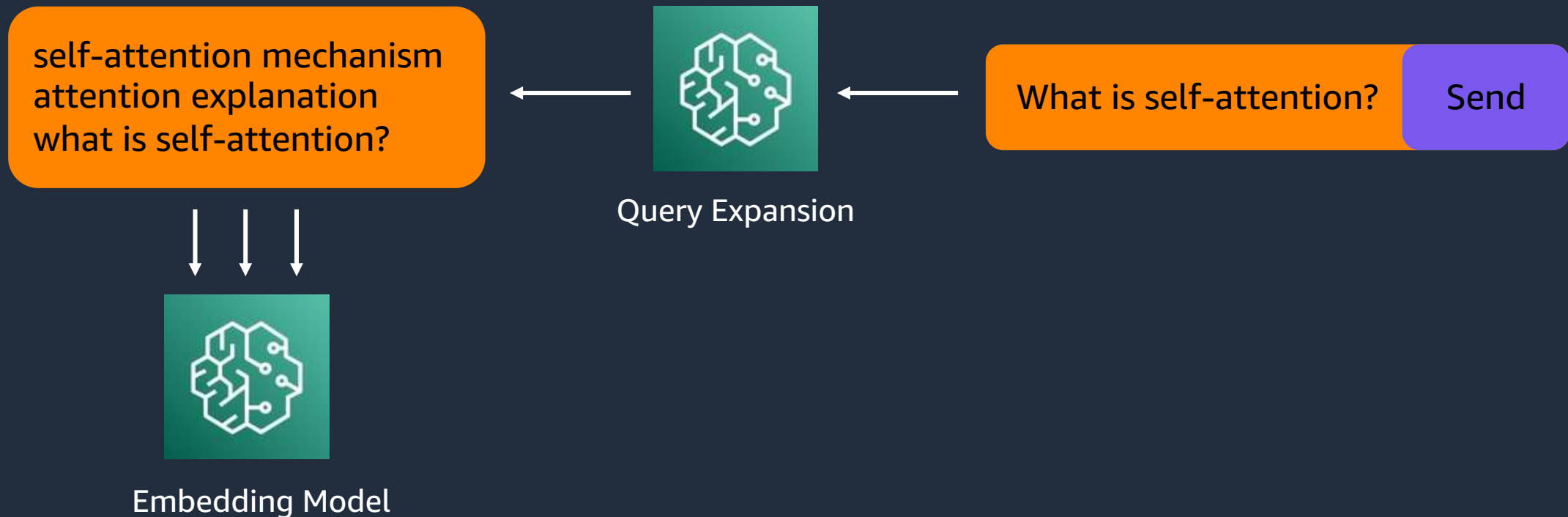
4.1. Simple Rewriting

Idea: reformulate user question into a better-structured search query



4.2. Query Expansion

Idea: reformulate user question into MULTIPLE search queries



Techniques to Improve Retrievers

#5. Document Reranking

5. Document Reranking

Idea: rerank retrieved documents (e.g., using LLM)

Why:

- Documents with the highest embedding similarity may not be most relevant
- LLM can rerank document chunks to better match the user query

Chunk 1

Chunk 2

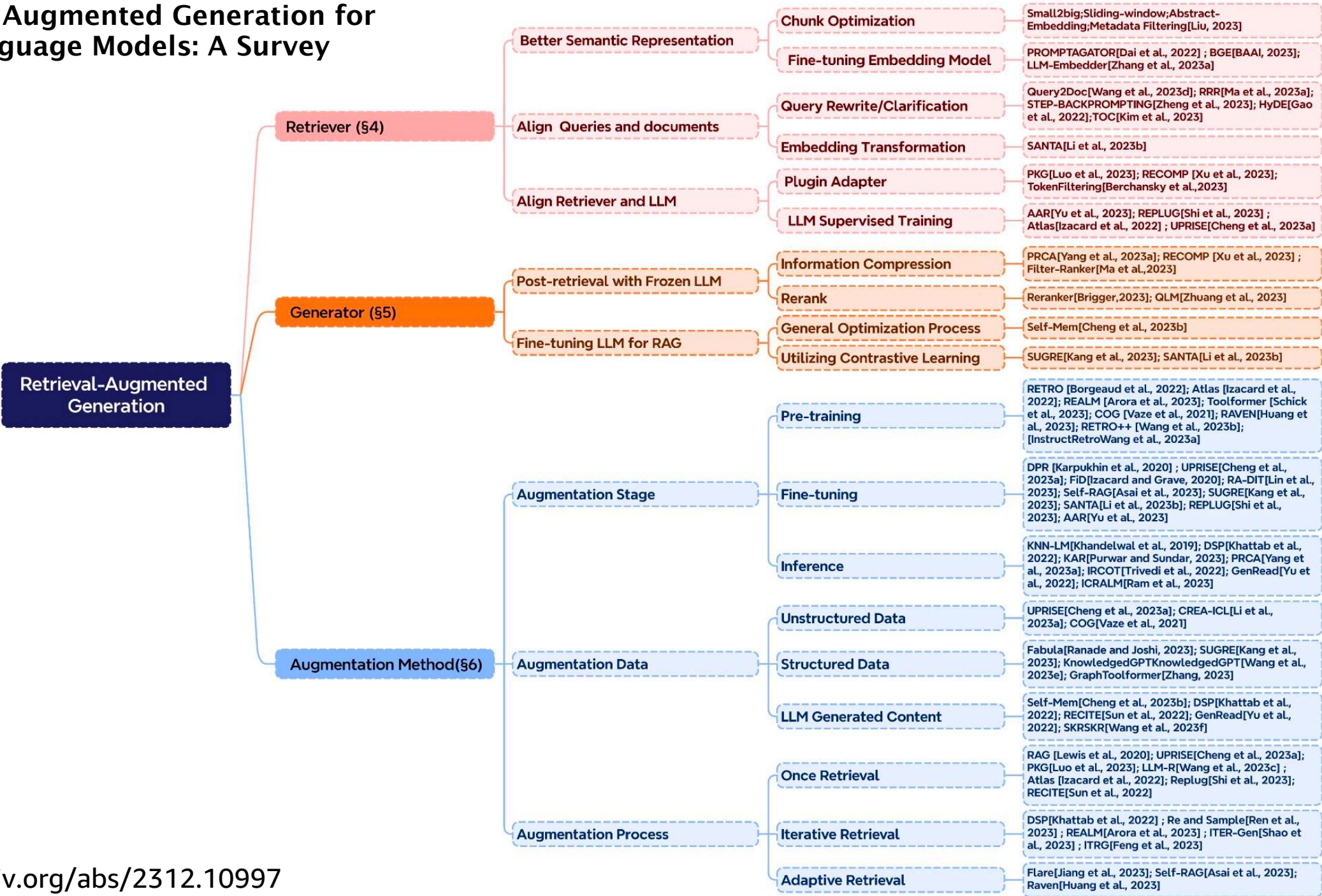
Chunk 3

5. Document Reranking

Idea: rerank retrieved documents (e.g., using LLM)



Retrieval-Augmented Generation for Large Language Models: A Survey





Thank You!

- **#1:** Small-to-Big Retrieval
- **#2:** Leveraging Meta-Data
- **#3:** Hybrid Search
- **#4:** Query Rewriting
- **#5:** Document Reranking

Nikita Kozodoi, PhD

