7 issues of RAG in production

And how to solve them

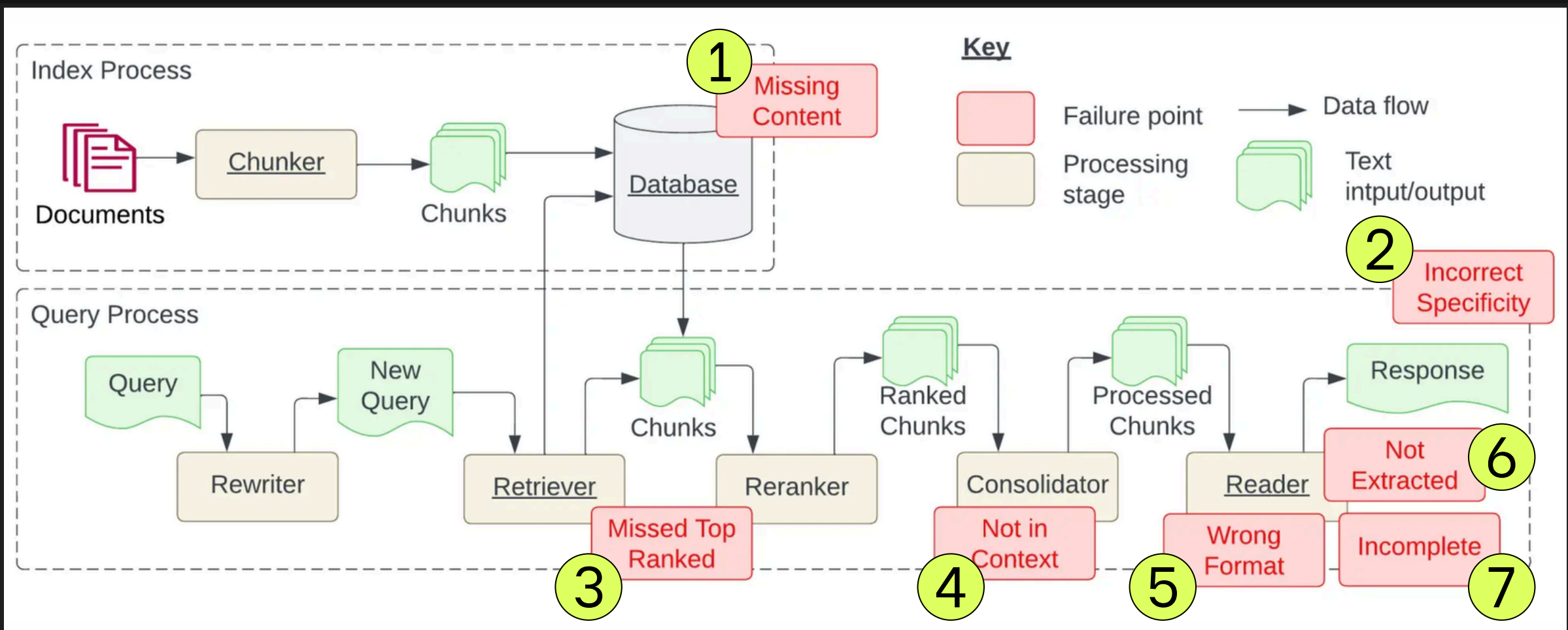
Joanna Stoffregen

co-founder & AI Product Lead
@Labsbit.ai

arXiv *Seven Failure Points When Engineering a Retrieval Augmented Generation System*

These are the most common **issues in RAG**

1. Missing Content
2. Incorrect Specificity
3. Missed top ranked

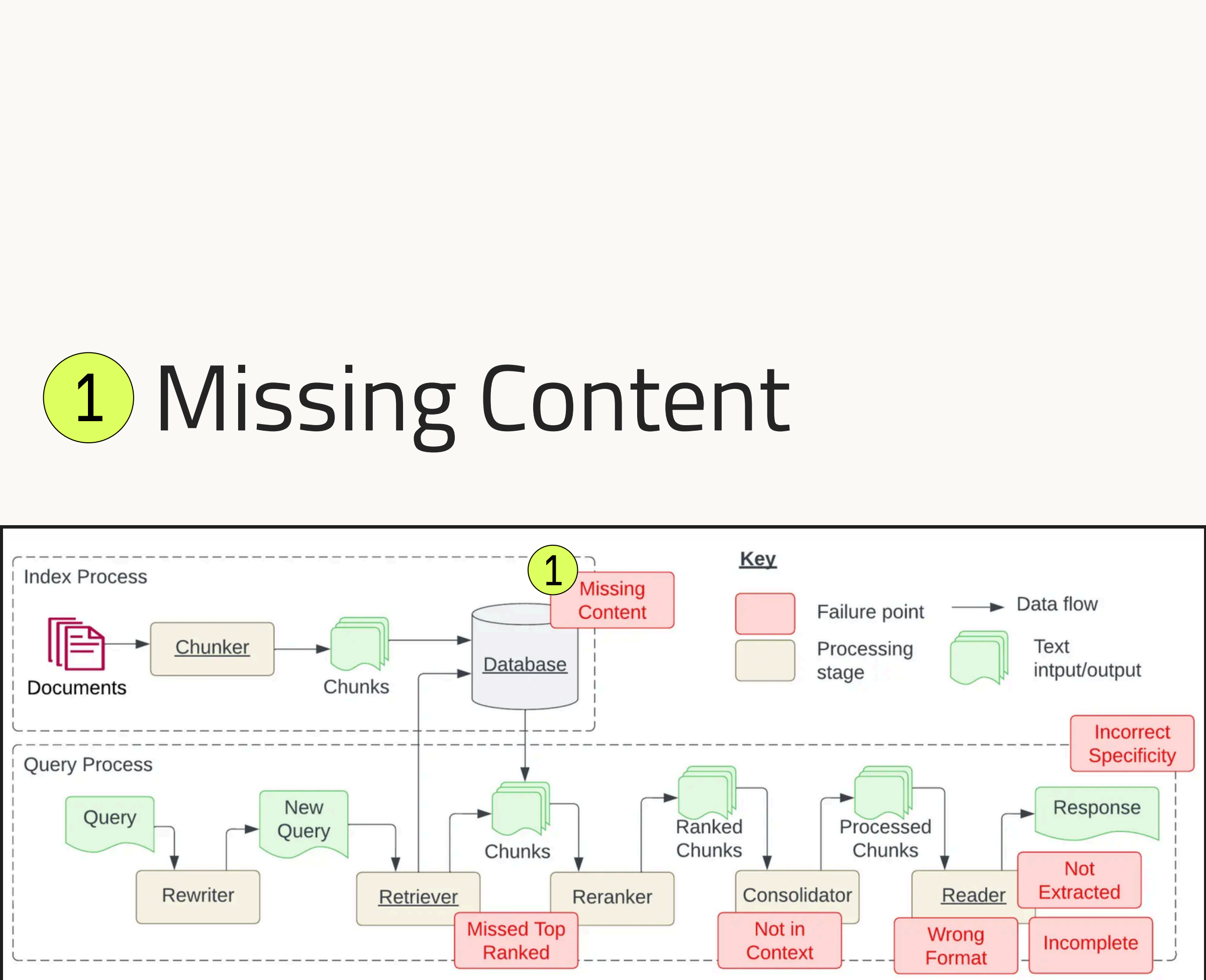


4. Not in context
6. Not extracted
5. Wrong Format
7. Incomplete

Let's look into what they mean

...and how to solve them

1 Missing Content



1 Missing Content

The problem

- This issue happens when context is missing from the knowledge base.

- The model provides an incorrect answer, rather than answering *"I don't know"*

Solutions

1. Clean the data

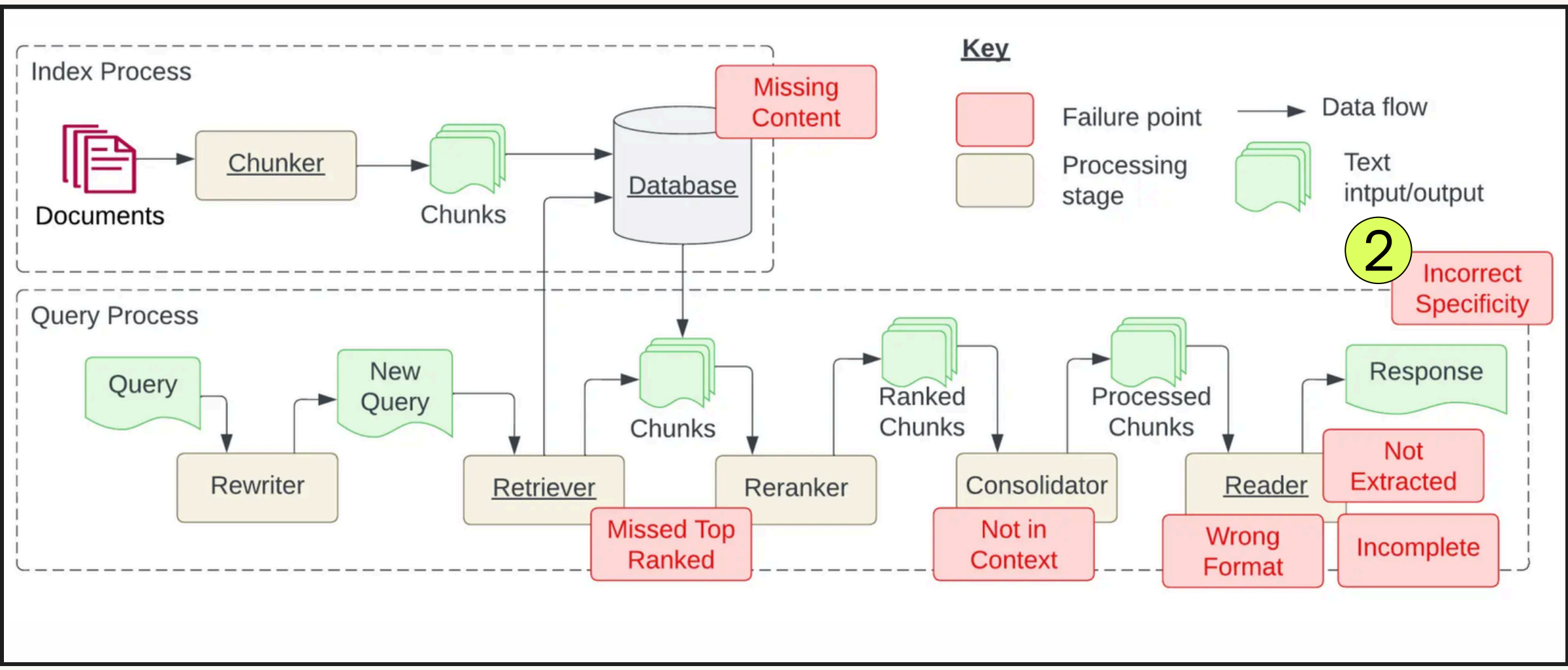
- Remove noise and irrelevant information
- Errors like spelling mistakes, typos, grammar etc
- Remove duplicates

Unstructured.io offers a set of cleaning functionalities

2. Better prompting

- Instruct the system *“Tell me you don’t know if you are not sure of the answer”*
- Not a guarantee - but worth trying!

2 Incorrect Specificity



2 Incorrect Specificity

The problem

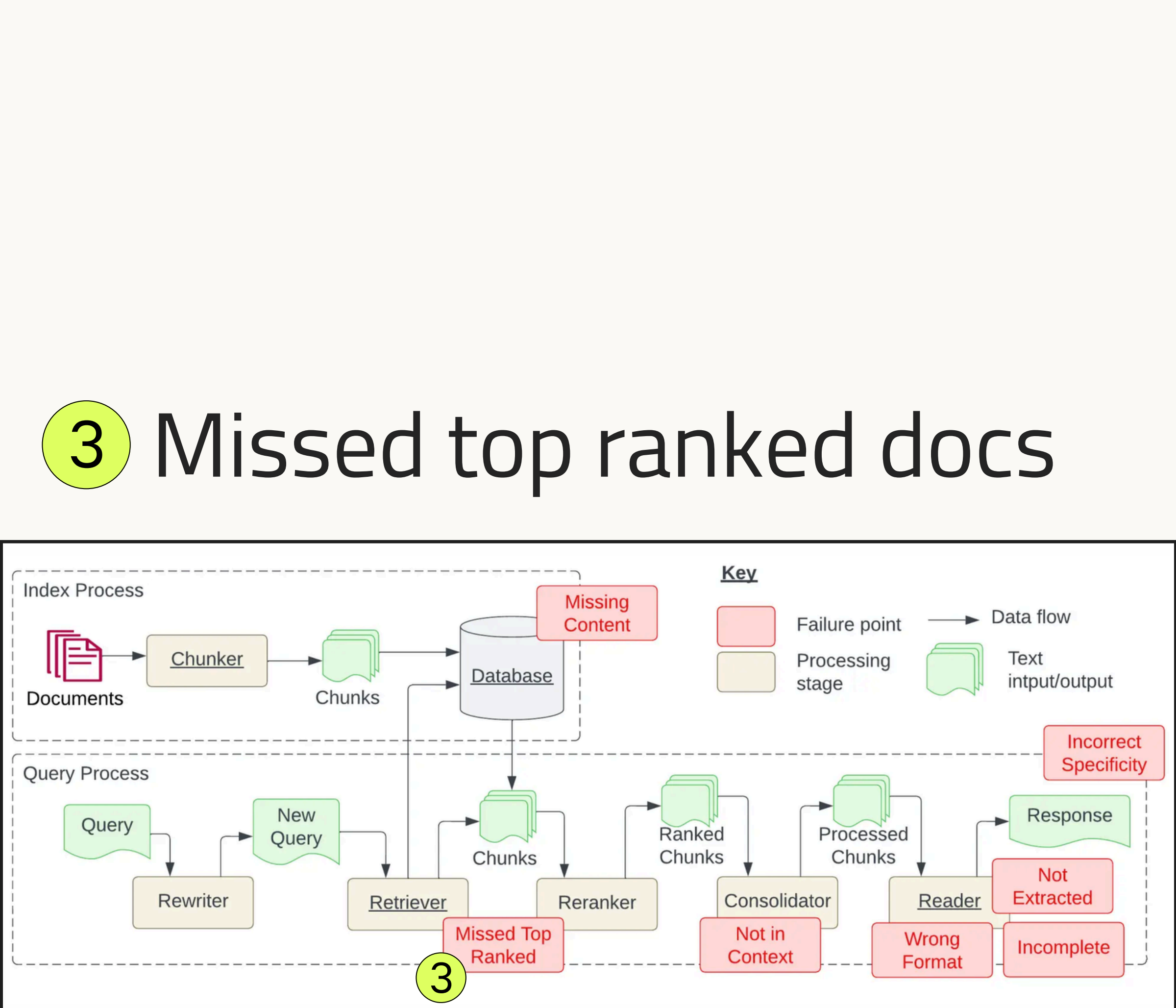
- The output is to vague and does not contain necessary details or specificity.
- It requires follow-up queries

Solutions

1. Try advanced retrieval strategies

- small-to-big retrieval
- sentence window retrieval
- recursive retrieval

3 Missed top ranked docs



3 Missed top ranked docs

The problem

- The system fails to give accurate responses because *“The answer to the question is in the document but did not rank highly enough to be returned to the user”.*

Solutions

1. Reranking

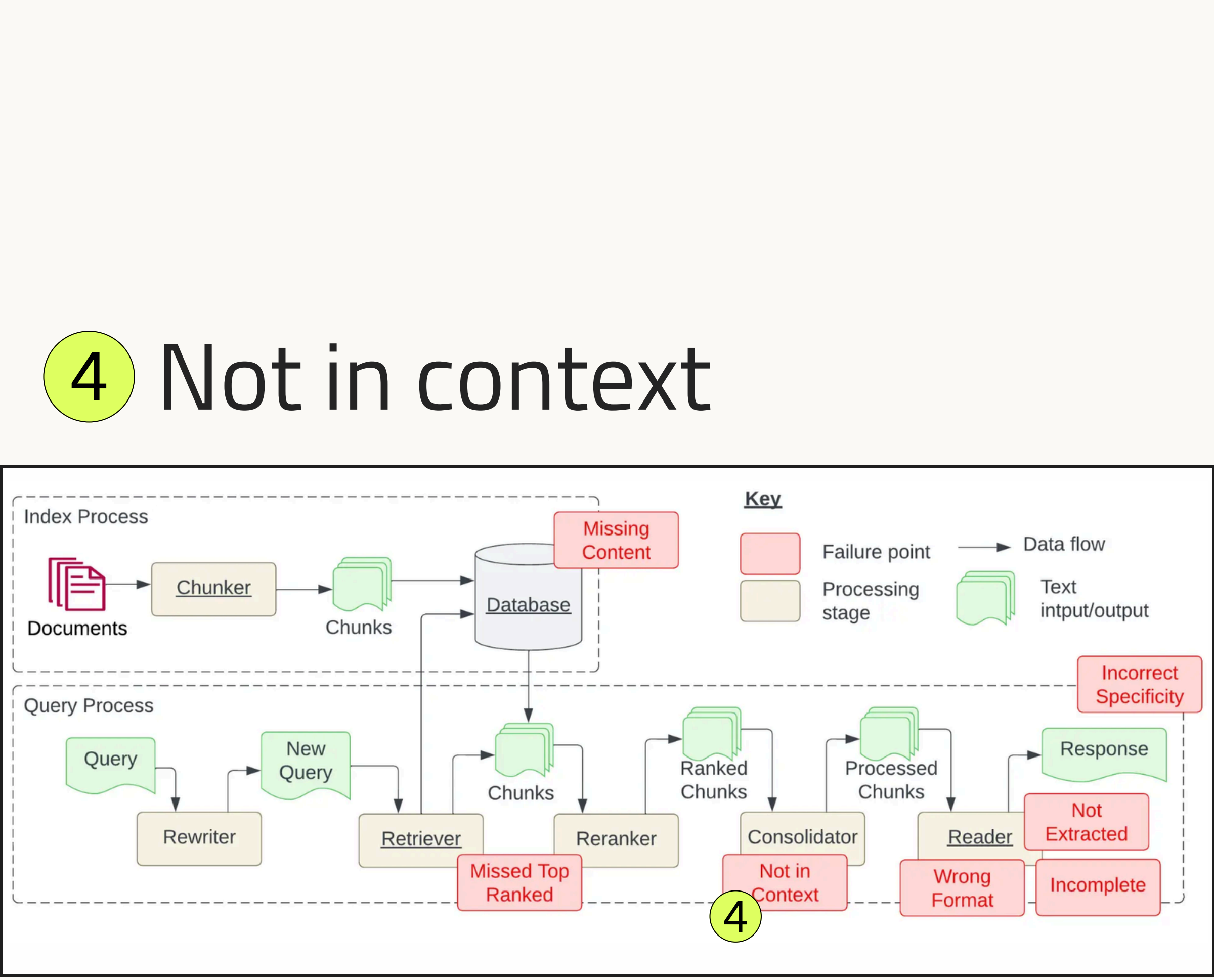
- Reranking retrieval results before sending them to the LLM can improved performance.

Check: Boosting RAG: Picking the Best Embedding & Reranker models

2. **Tune** chunk_size **and** similarity_top_k **hypermarameters**

Check: Automating Hyperparameter Tuning with LlamaIndex

4 Not in context



4 Not in context

The problem

- “Documents with the answer were retrieved from the database but did not make it into the context for generating an answer.
- This occurs when many documents are returned from the database, and a consolidation process takes place to retrieve the answer”.

Solutions

1. Experiment with different retrieval strategies

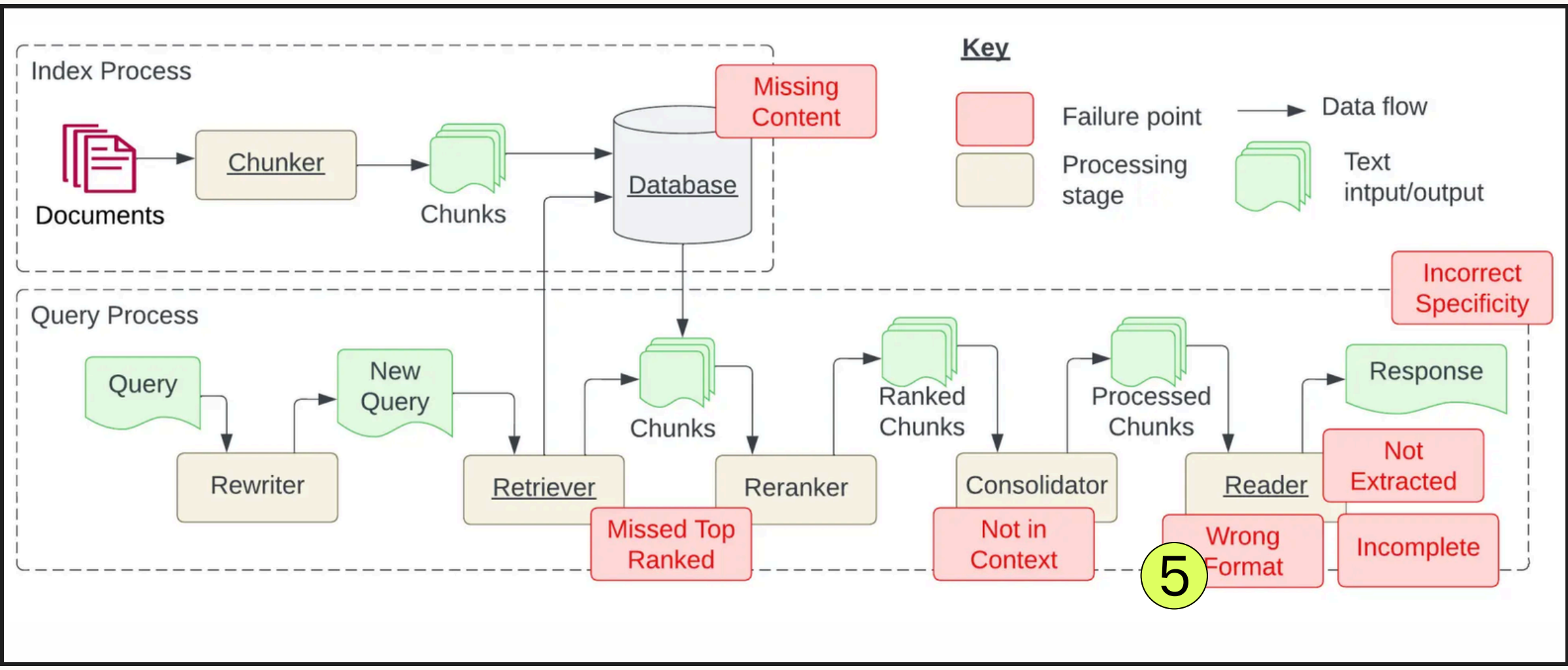
- Basic retrieval from each index
- Advanced retrieval and search
- Auto-Retrieval
- Knowledge Graph Retrievers
- Composed/Hierarchical Retrievers

Check: The retrievers module guide

2. Finetune embeddings

LlamaIndex has a step-by-step guide on finetuning an open-source embedding model

5 Wrong Format



5 Wrong Format

The problem

- The output is in the wrong format.
- For example you get a text instead of a table

Solutions

1. Better instructions / prompting

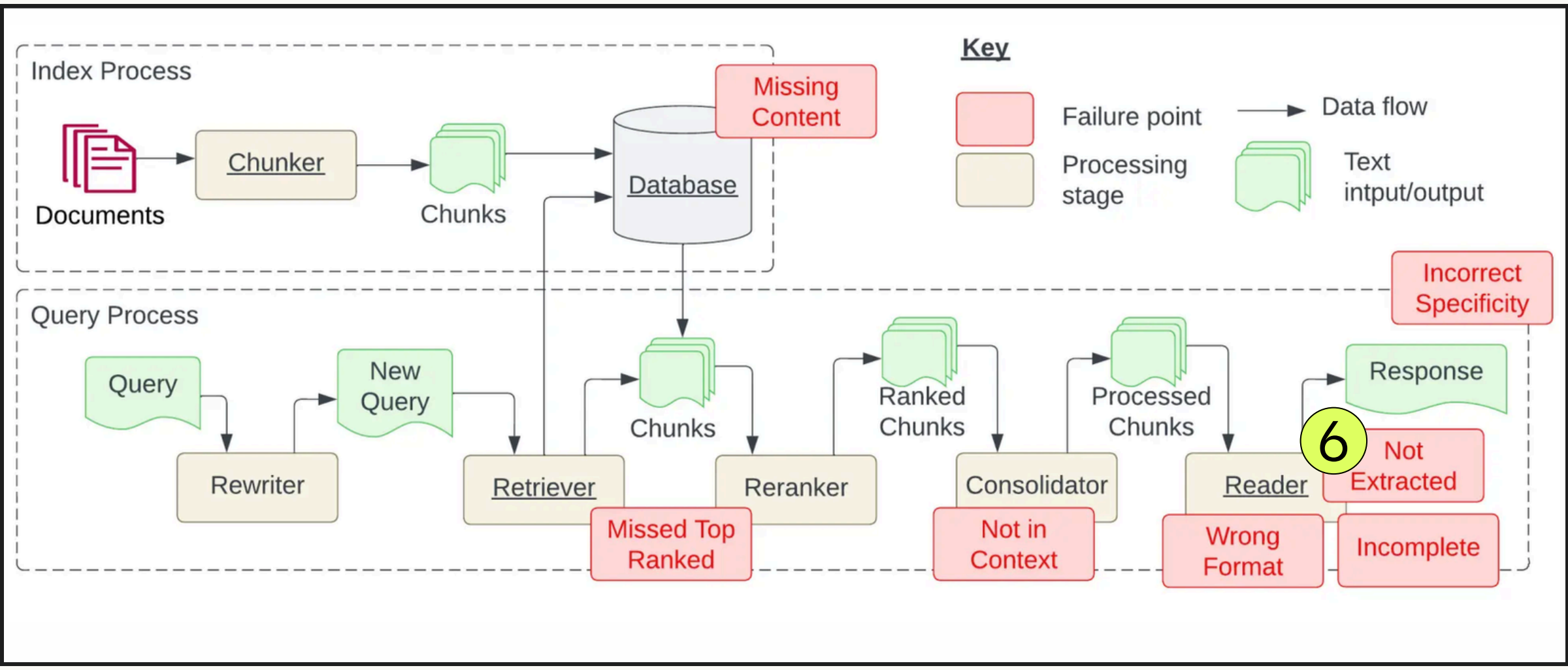
- Simplify the request and clarify the instructions
- Give examples and ask follow up questions

2. Output parsing

- Provide formatting instructions and parsing for LLM outputs

Check output parsing modules by Guardrails and LangChain

6 Not extracted



6 Not extracted

The problem

- The system struggles to extract the correct answer from the provided context
- *"This occurs when there is too much noise or contradicting information in the context".*

Solutions

1. Clean your the data

2. Use Prompt compression technique

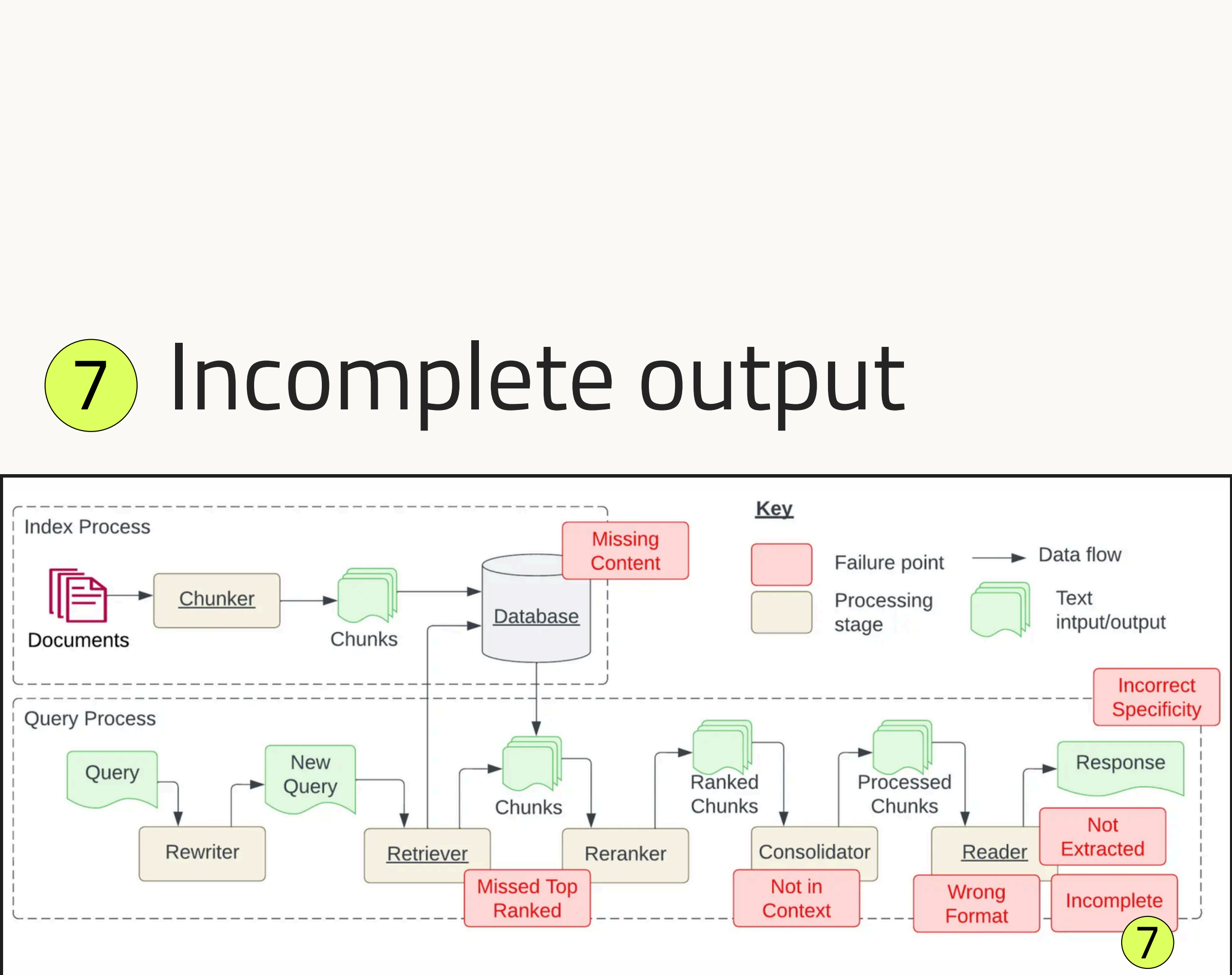
- Compress context after the retrieval step before feeding it into the LLM.

Implement LongLLMLingua as a node postprocessor

3. LongContextReorder

- Performance increased when important data is positioned at the start or conclusion of the input context.
- LongContextReorder address the "lost in the middle" problem by re-ordering the retrieved nodes

7 Incomplete output



7 Incomplete output

The problem

- Systems replies with partial information even though the information is being present and accessible within the context.

Solutions

1. Query transformations

- Naive RAG performs poorly when asked comparison questions like: *“What are the main aspects discussed in documents A, B, and C?”*
- To improve reasoning capability is to add a query understanding layer before actually querying the vector store

- **Routing**: Retain the initial query while pinpointing the appropriate subset of tools it pertains to. Then, designate these tools as the suitable options.
- **Query-Rewriting**: Maintain the selected tools, but reformulate the query in multiple ways to apply it across the same set of tools.
- **Sub-Questions**: Break down the query into several smaller questions, each targeting different tools as determined by their metadata.
- **ReAct Agent Tool Selection**: Based on the original query, determine which tool to use and formulate the specific query to run on that tool.

Source: towardsdatascience.com/12-rag-pain-points-and-proposed-solutions-43709939a28c#cea4

The solutions have been adapted from the folks at Llamaindex from the video linked bellow

I highly suggest you to check it out, as they discuss in much more detail the solutions

LlamaIndex

LlamaIndex Sessions: 12 RAG Pain Points and Solutions

I'm Joanna from Labsbit.ai

An AI Product Development
& Automation Company

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