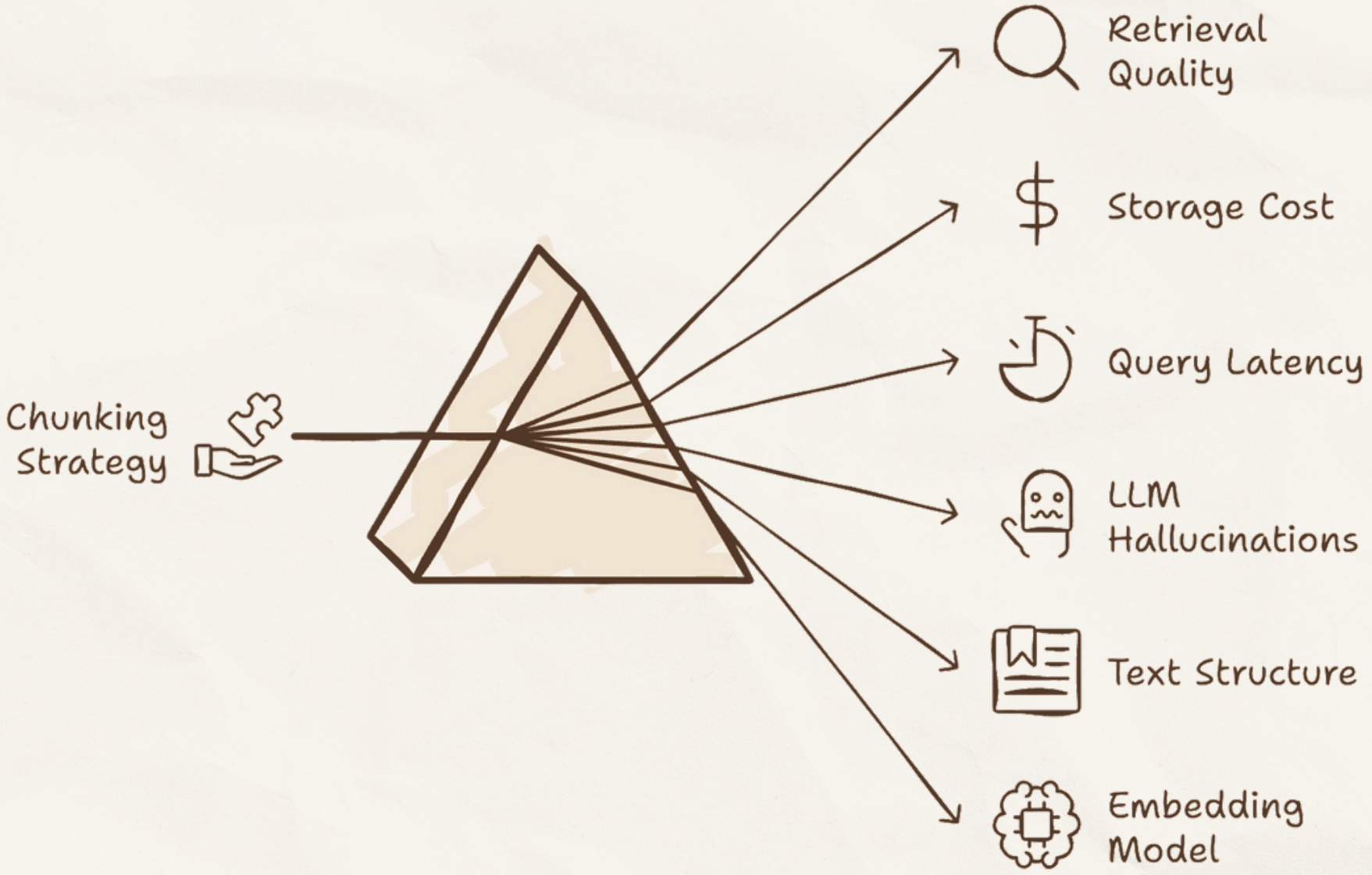
6 TYPES OF CHUNKING IN RAG

Bhavishya Pandit

INTRODUCTION

Chunking involves breaking down texts into smaller, manageable pieces called "chunks." Each chunk becomes a unit of information that is vectorized and stored in a database, fundamentally shaping the efficiency and effectiveness of natural language processing tasks.



Impact of chunking

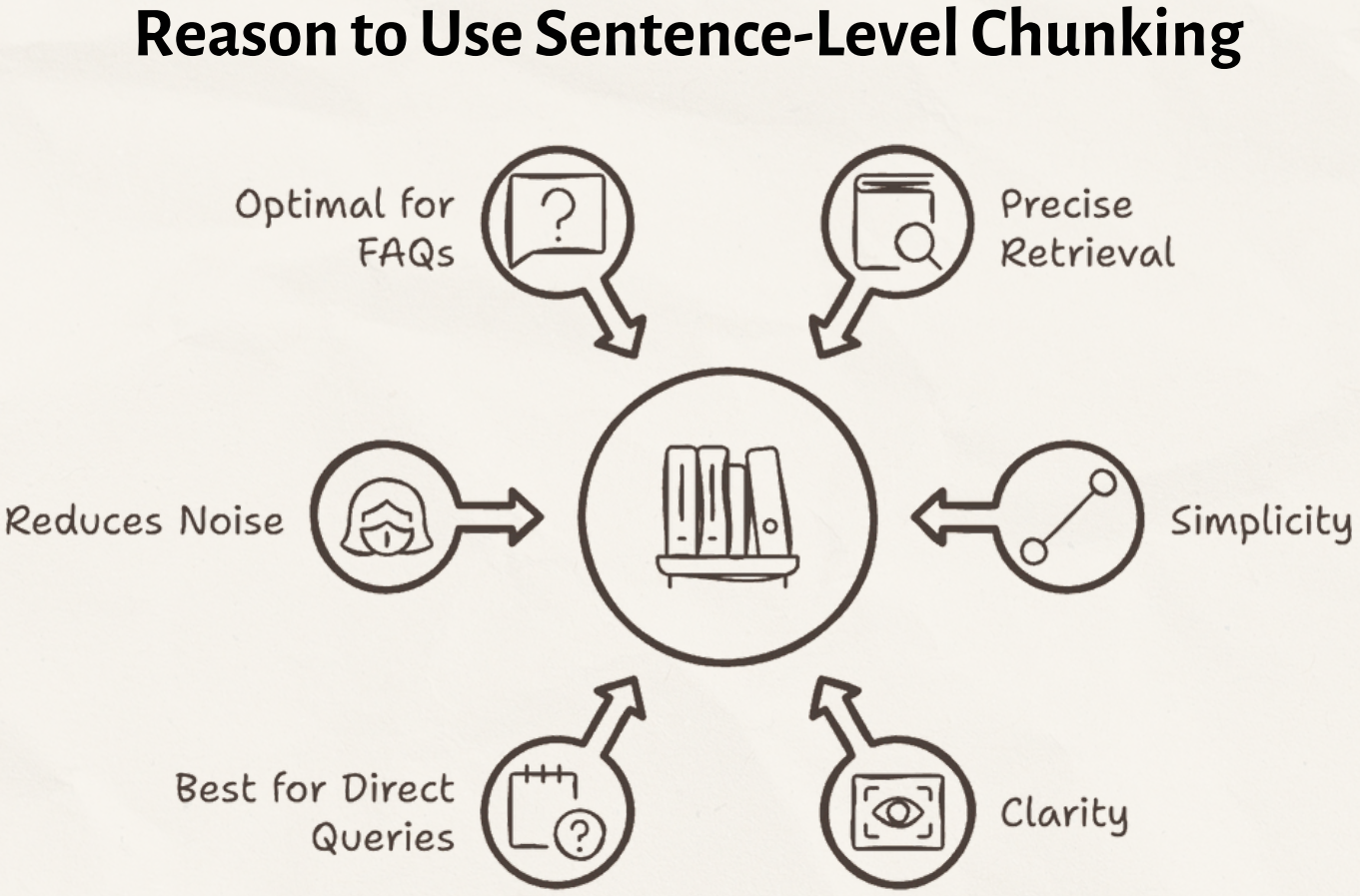
Chunking plays a central role in various aspects of RAG systems, exerting influence not only on retrieval quality but also on response.

Factors influencing chunking

We understand the importance of taking chunking seriously, but what factors influence it? A better understanding of these parameters will enable us to select an appropriate strategy.

SENTENCE-LEVEL CHUNKING

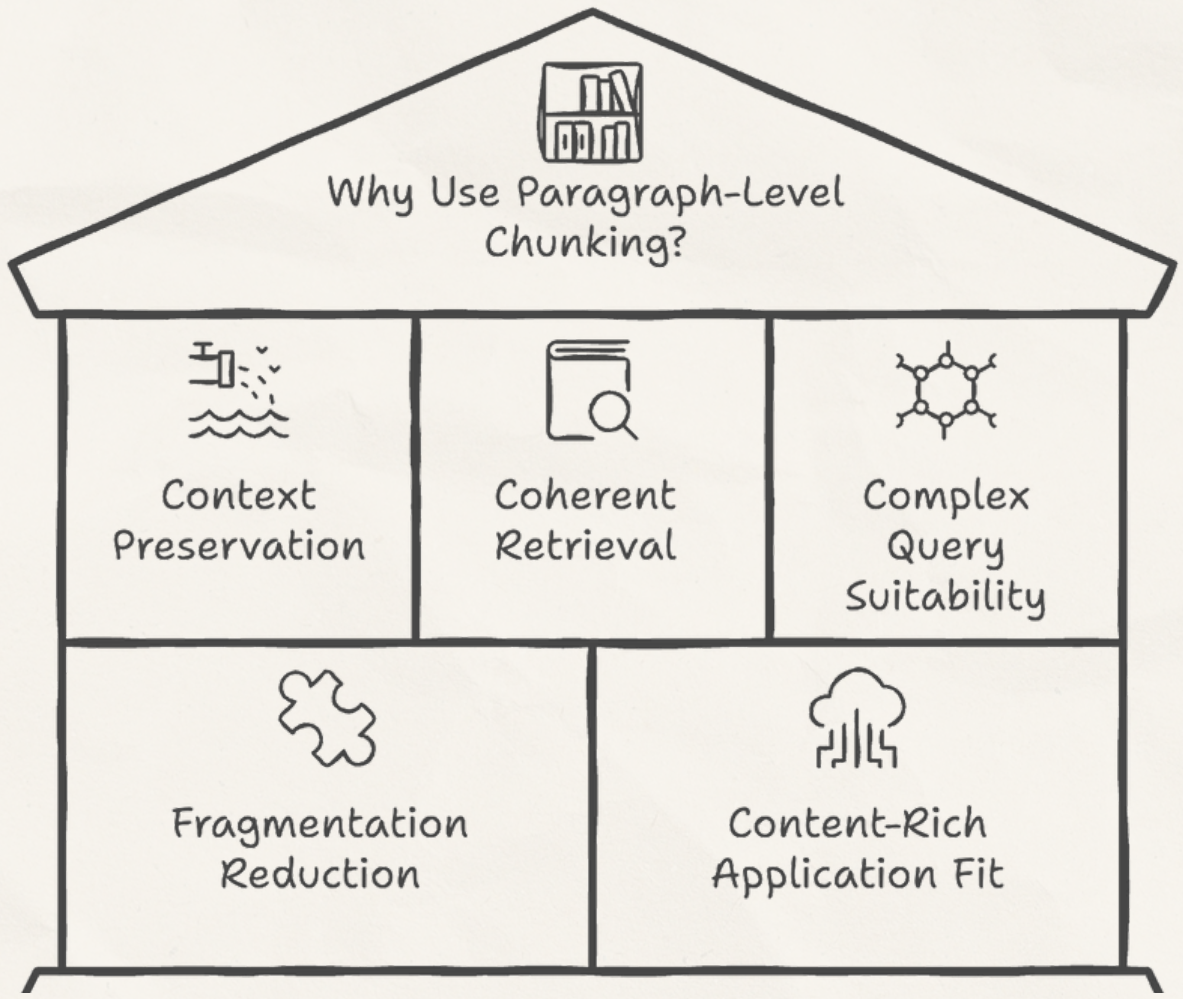
It involves breaking text into individual sentences, where each sentence is treated as a separate chunk.



- **Advantages:**
  - Excellent for retrieving precise, factual information.
  - Simplifies retrieval when each sentence conveys an independent idea.
- **Disadvantages:**
  - May lose important context if ideas are split across sentences.
  - Can lead to fragmented or disjointed retrieval if concepts span multiple sentences
- **Use Cases:**
  - Factual queries: Ideal when users ask simple questions that require a specific answer
  - FAQ databases: Useful when each answer is concise and independent

PARAGRAPH-LEVEL CHUNKING

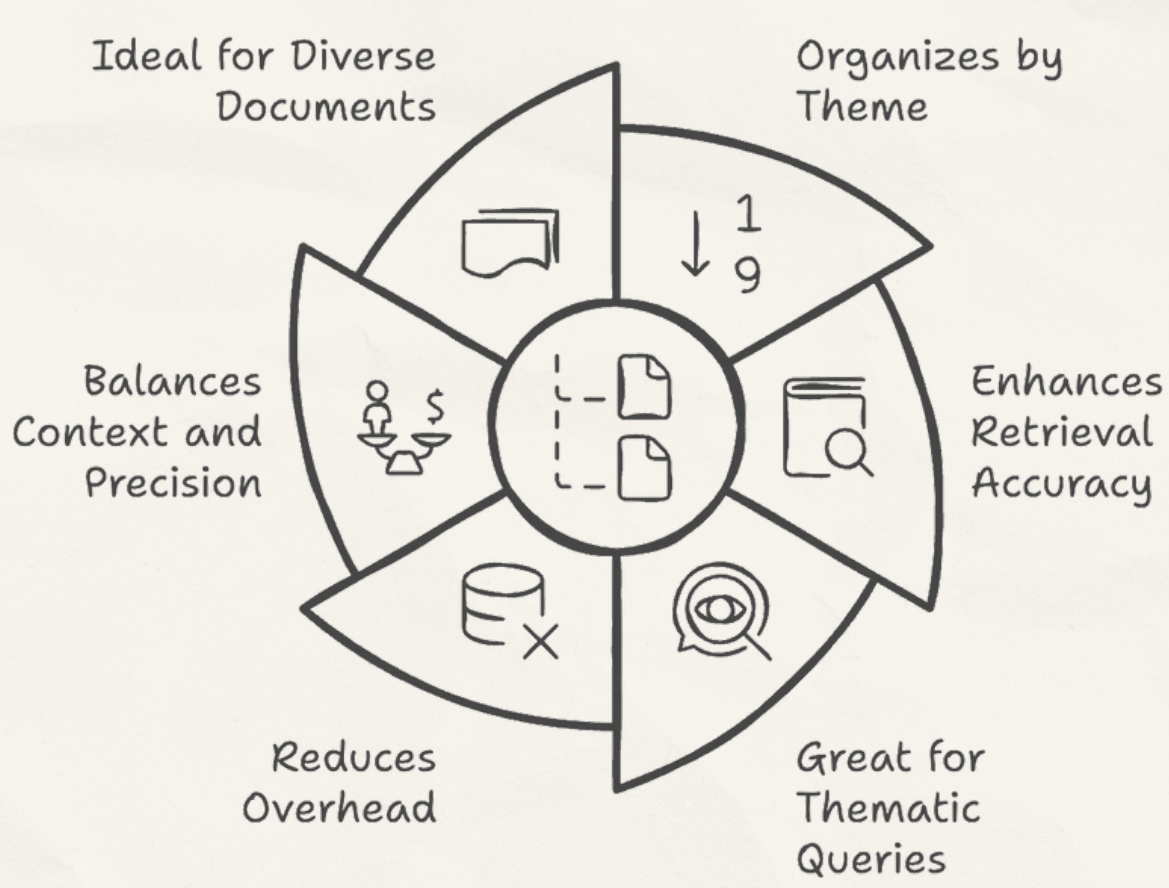
This method involves dividing text into paragraphs, where each paragraph becomes a single chunk, capturing several related sentences.



- **Advantages:**
  - Maintains context and flow, as related ideas are kept together.
  - Reduces the need for multiple chunks to answer complex questions.
- **Disadvantages:**
  - Larger chunks may push the limits of the LLM's context window, reducing efficiency.
  - Could lead to irrelevant details being retrieved along with the useful information.
- **Use Cases:**
  - Complex queries: Best for questions that require multi-sentence answers or contextual understanding.
  - Content-rich applications: Useful in research or in-depth knowledge retrieval.

TOPIC-BASED CHUNKING

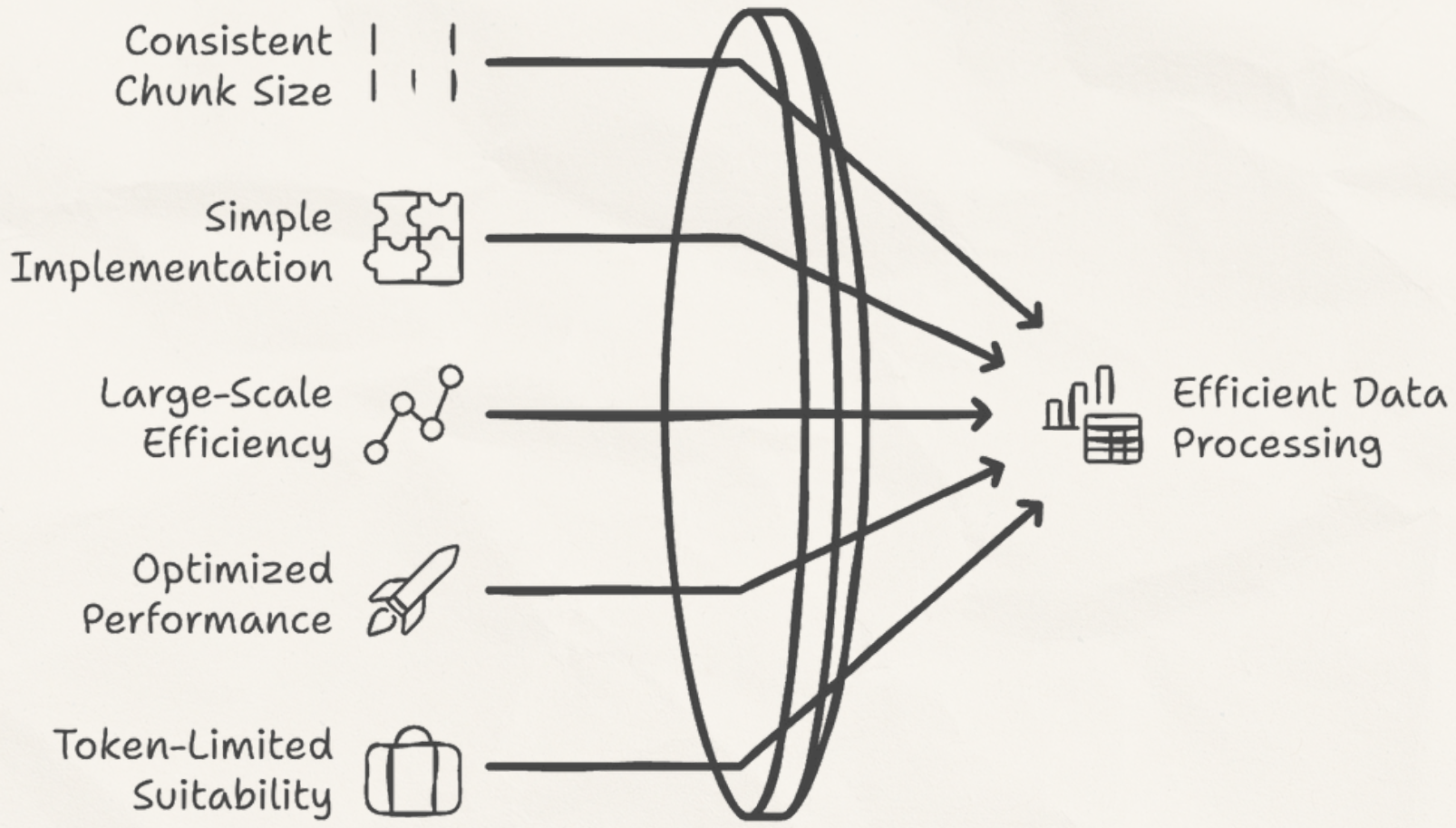
Topic-based chunking involves dividing text based on the theme or subject matter, grouping sentences or paragraphs that revolve around a specific topic or concept into one chunk.



- **Advantages:**
  - Provides comprehensive and thematic responses, especially for broad questions.
  - Reduces the number of chunks when dealing with text that covers diverse topics in one document.
- **Disadvantages:**
  - Requires additional preprocessing to define and group topics accurately.
  - Could result in losing granularity if topics aren't carefully distinguished.
- **Use Cases:**
  - Thematic knowledge bases: Perfect for applications like legal or medical databases where information is organized by topic.
  - User-generated content: Beneficial when users ask multi-faceted or thematic questions.

FIXED-SIZE CHUNKING

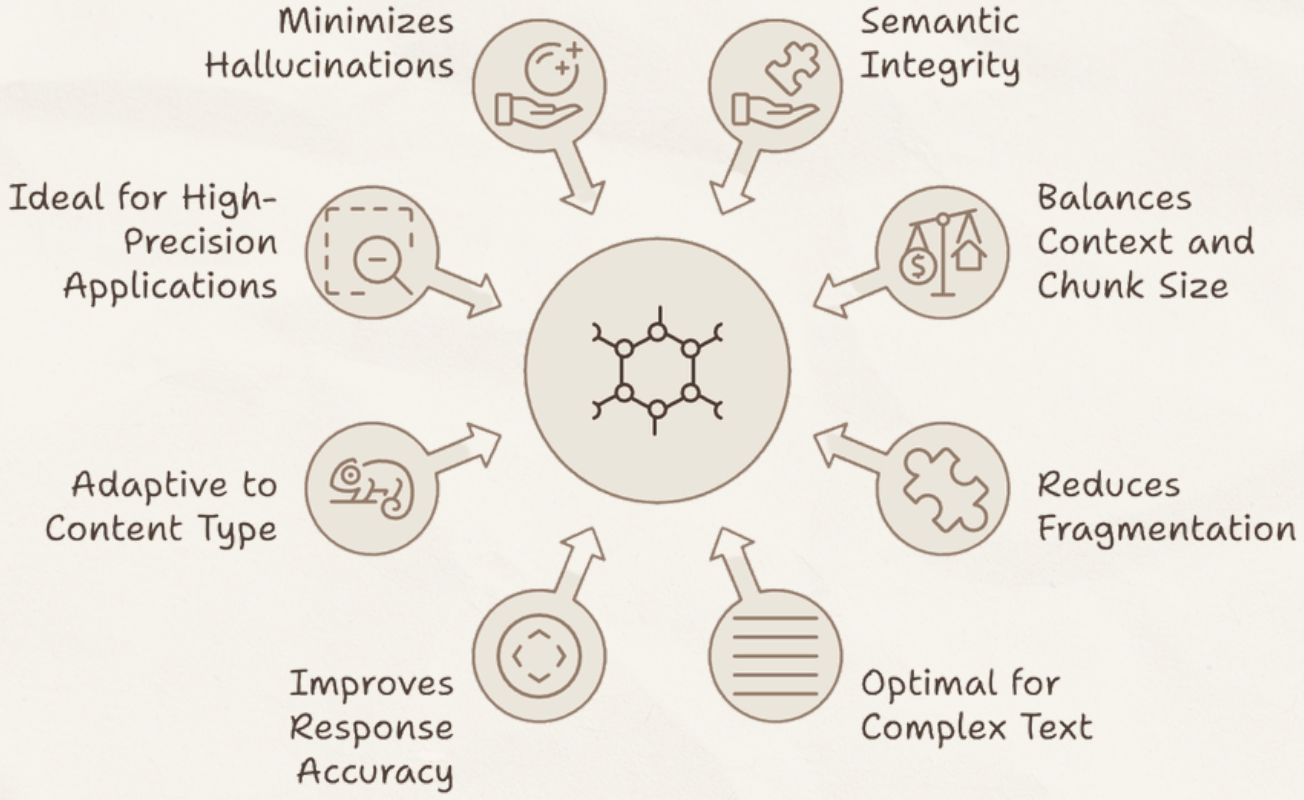
Fixed-size chunking divides text into chunks of a predetermined length, based on word count, character count, or token count, without considering natural boundaries like sentences or paragraphs



- **Advantages:**
  - Simple to implement and ensures consistency in chunk size.
  - Predictable storage and retrieval, making it easy to manage data at scale.
- **Disadvantages:**
  - May cut off sentences, resulting in incomplete ideas.
  - Retrieval could lack relevance if the chunks do not align with meaningful content boundaries.
- **Use Cases:**
  - Large-scale document processing: Suitable when uniformity is a priority, such as in processing massive text corpora.
  - Token-limit environments: Ideal for systems constrained by input/output token limits.

CONTEXT-AWARE CHUNKING

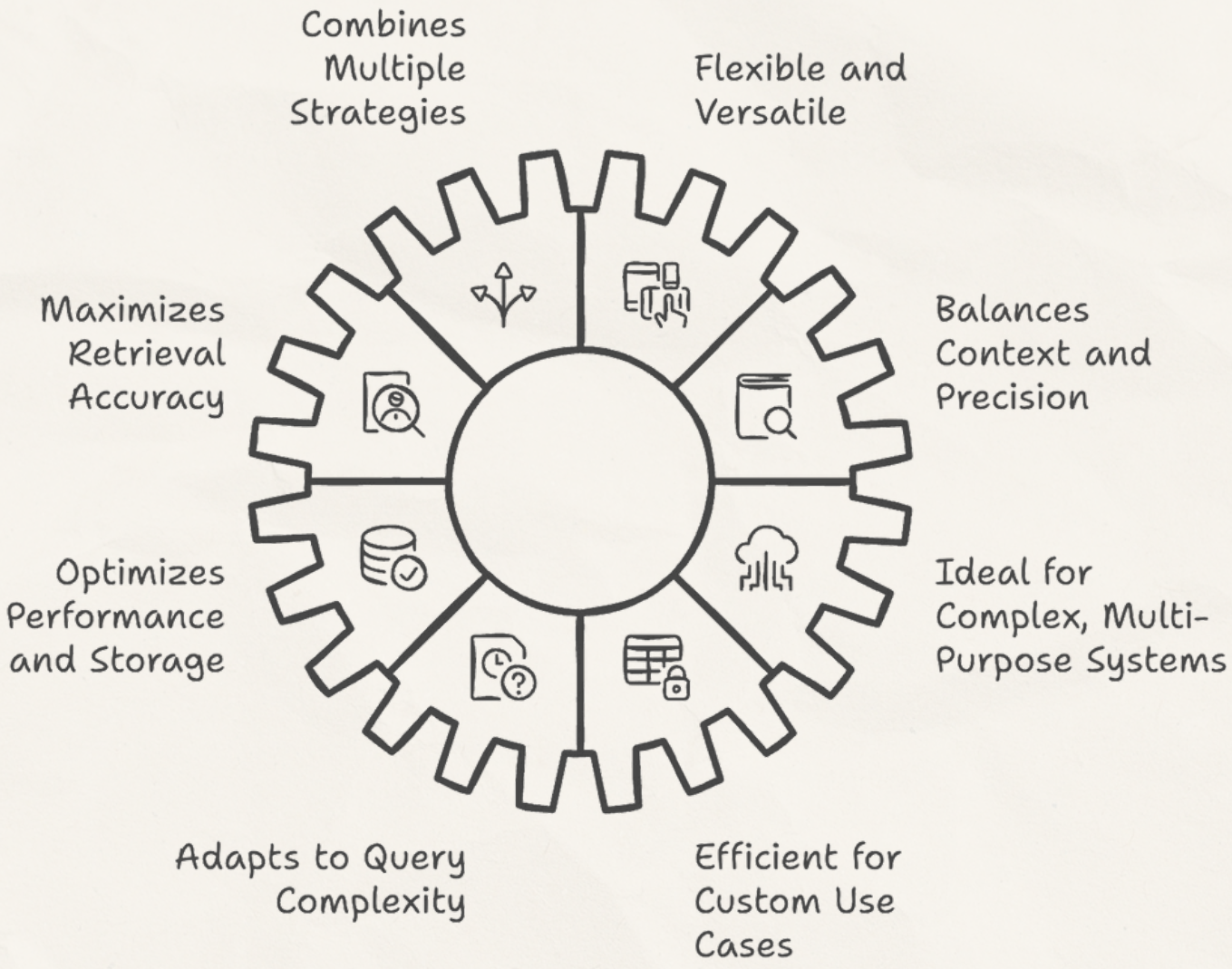
Context-aware chunking involves breaking text into chunks based on the semantic meaning and structure of the content, ensuring that each chunk contains complete, coherent thoughts or ideas.



- **Advantages:**
  - Retains a balance between coherence and size, preserving meaningful context.
  - Adjusts chunk size dynamically based on the content's complexity and importance
- **Disadvantages:**
  - More complex to implement, requiring advanced algorithms or AI models to define appropriate chunks.
  - May need more processing time to determine the chunk boundaries.
- **Use Cases:**
  - High-quality LLM responses: Perfect for customer service or support chatbots where understanding context is critical.
  - Complex text processing: Used in academic research, where maintaining semantic integrity is crucial

HYBRID CHUNKING

Hybrid chunking combines two or more chunking techniques (such as sentence-level, paragraph-level, or topic-based) to create chunks that are tailored to specific use cases, balancing granularity, context, and retrieval needs



- **Advantages:**
  - Offers flexibility to handle varied data types and retrieval needs.
  - Can optimize both precision and recall by using different methods for different parts of the text.
- **Disadvantages:**
  - Requires careful tuning to prevent inconsistencies or mismatches in chunk types.
  - May involve greater system complexity and resource use
- **Use Cases:**
  - Multi-faceted applications: Ideal for use cases like knowledge graphs or document databases where text types vary widely.
  - Custom retrieval systems: Can be tailored to optimize retrieval for specific user interactions.

WHAT OTHER TYPES OF CHUNKING DO YOU KNOW?

Follow to stay updated on AI/ML

SAVE

LIKE

SHARE