Top 13 Advanced RAG Techniques

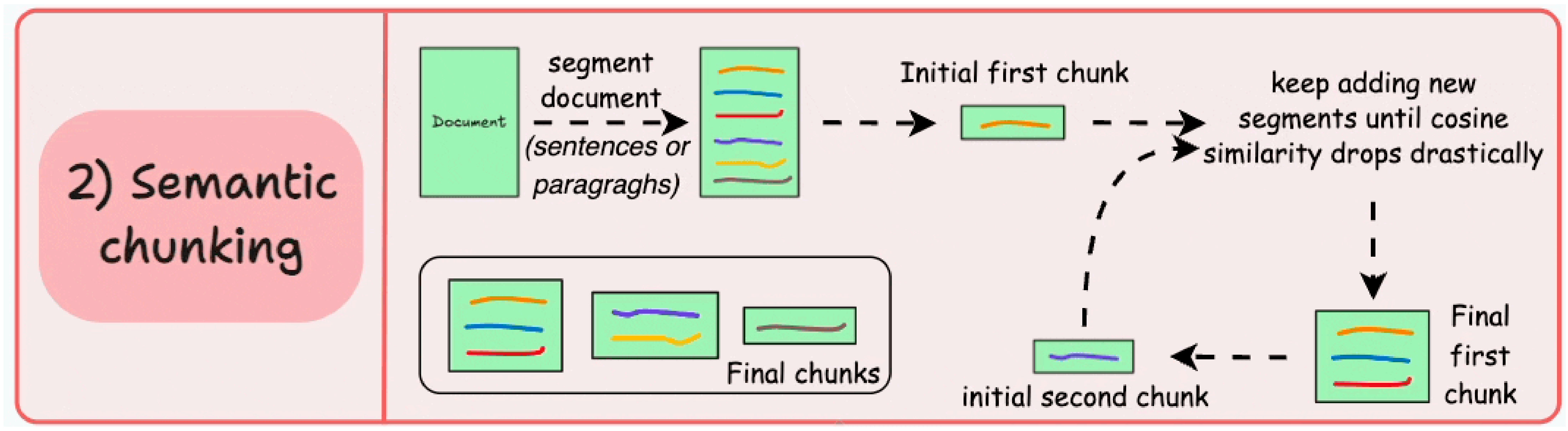
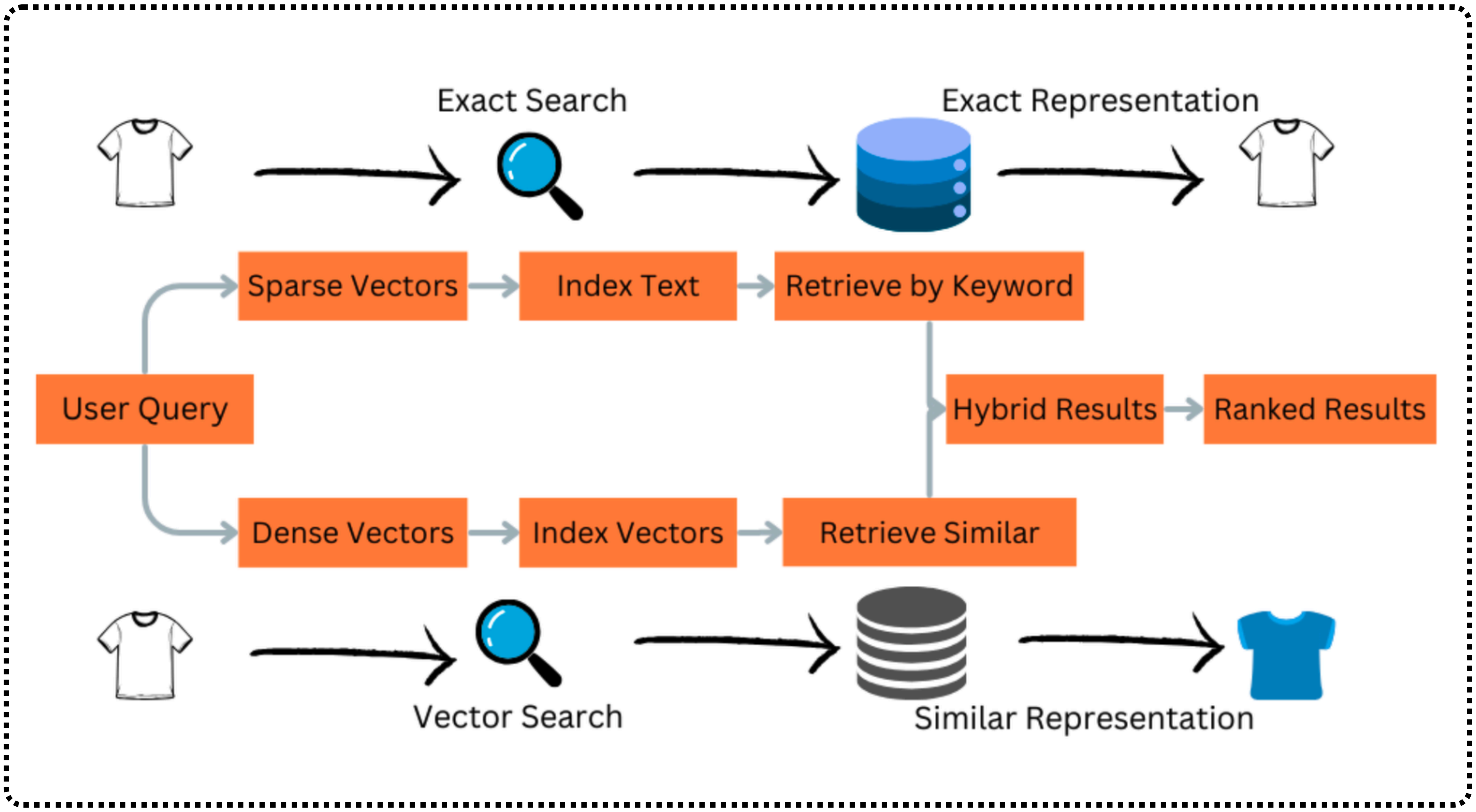


Image Credits: DailydoseofDS



Indexing and Chunking: Building a Strong Foundation

Good indexing is essential for any RAG system. The first step involves how we bring in, break up, and store data. Let's explore methods to index data, focusing on indexing and chunking text and using metadata.

1. HNSW: Hierarchical Navigable Small Worlds

Hierarchical Navigable Small Worlds (HNSW) is an effective algorithm for finding similar items in large datasets. It helps in quickly locating approximate nearest neighbors (ANN) by using a structured approach based on graphs.

- **Proximity Graph**: HNSW builds a graph where each point connects to nearby points. This structure allows for efficient searching.
- **Hierarchical Structure**: The algorithm organizes points into multiple layers. The top layer connects distant points, while lower layers connect closer points. This setup speeds up the search process.
- **Greedy Routing**: HNSW uses a greedy method to find neighbors. It starts at a high-level point and moves to the nearest neighbor until it reaches a local minimum. This method reduces the time needed to find similar items.

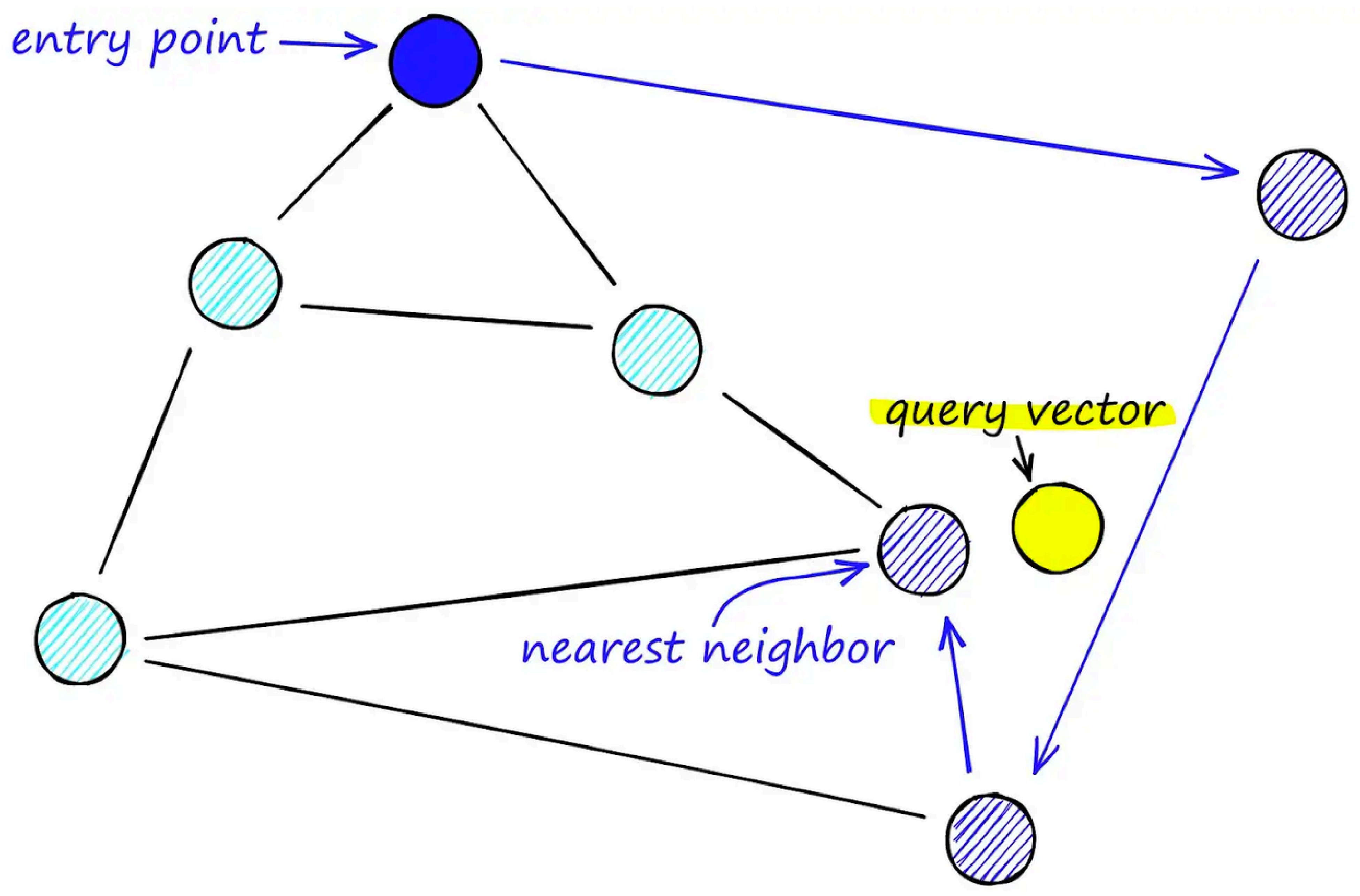
How does HNSW work?

The working of HNSW includes several key components:

- **Input Layer**:
  - Each data point is represented as a vector in a high-dimensional space.
- **Graph Construction**:
  - Nodes are added to the graph one at a time.
  - Each node is assigned to a layer based on a probability function. This function decides how likely a node is to be placed in a higher layer.
  - The algorithm balances the number of connections and the speed of searches.
- **Search Process**:
  - The search starts at a specific entry point in the top layer.
  - The algorithm moves to the nearest neighbor at each step.
  - Once it reaches a local minimum, it shifts to the next lower layer and continues searching until it finds the closest point in the bottom layer.

- Parameters:
    - **M**: The number of neighbors connected to each node.
    - **efConstruction**: This parameter affects how many neighbors the algorithm considers when building the graph.
    - **efSearch**: This parameter influences the search process, determining how many neighbors to evaluate.

HNSW's design allows it to find similar items quickly and accurately. This makes it a strong choice for tasks that require efficient searches in large datasets.



The image depicts a simplified HNSW search: starting at the “entry point” (blue), the algorithm navigates the graph towards the “query vector” (yellow). The “nearest neighbor” (striped) is identified by traversing edges based on proximity. This illustrates the core concept of navigating a graph for efficient approximate nearest neighbor search.

Semantic Chunking

This approach divides text based on meaning, not just fixed sizes. Each chunk represents a coherent piece of information. We calculate the cosine distance between sentence embeddings.

If two sentences are semantically similar (below a threshold), they go in the same chunk. This creates chunks of different lengths based on the content’s meaning.

- **Pros**: Creates more coherent and meaningful chunks, improving retrieval.
- **Cons**: Requires more computation (using a BERT-based encoder).

Language Model-Based Chunking

This advanced method uses a language model to create complete statements from text. Each chunk is semantically whole. A language model (e.g., a 7-billion parameter model) processes the text. It breaks it into statements that make sense on their own.

The model then combines these into chunks, balancing completeness and context. This method is computationally heavy but offers high accuracy.

- Pros: Adapts to the nuances of the text and creates high-quality chunks.
- Cons: Computationally expensive; may need fine-tuning for specific uses.

For more information, you can visit this article

Advanced RAG

Top 13 Advanced RAG Techniques for Your Next Project

Advanced RAG techniques to enhance retrieval, reduce hallucinations & improve resp