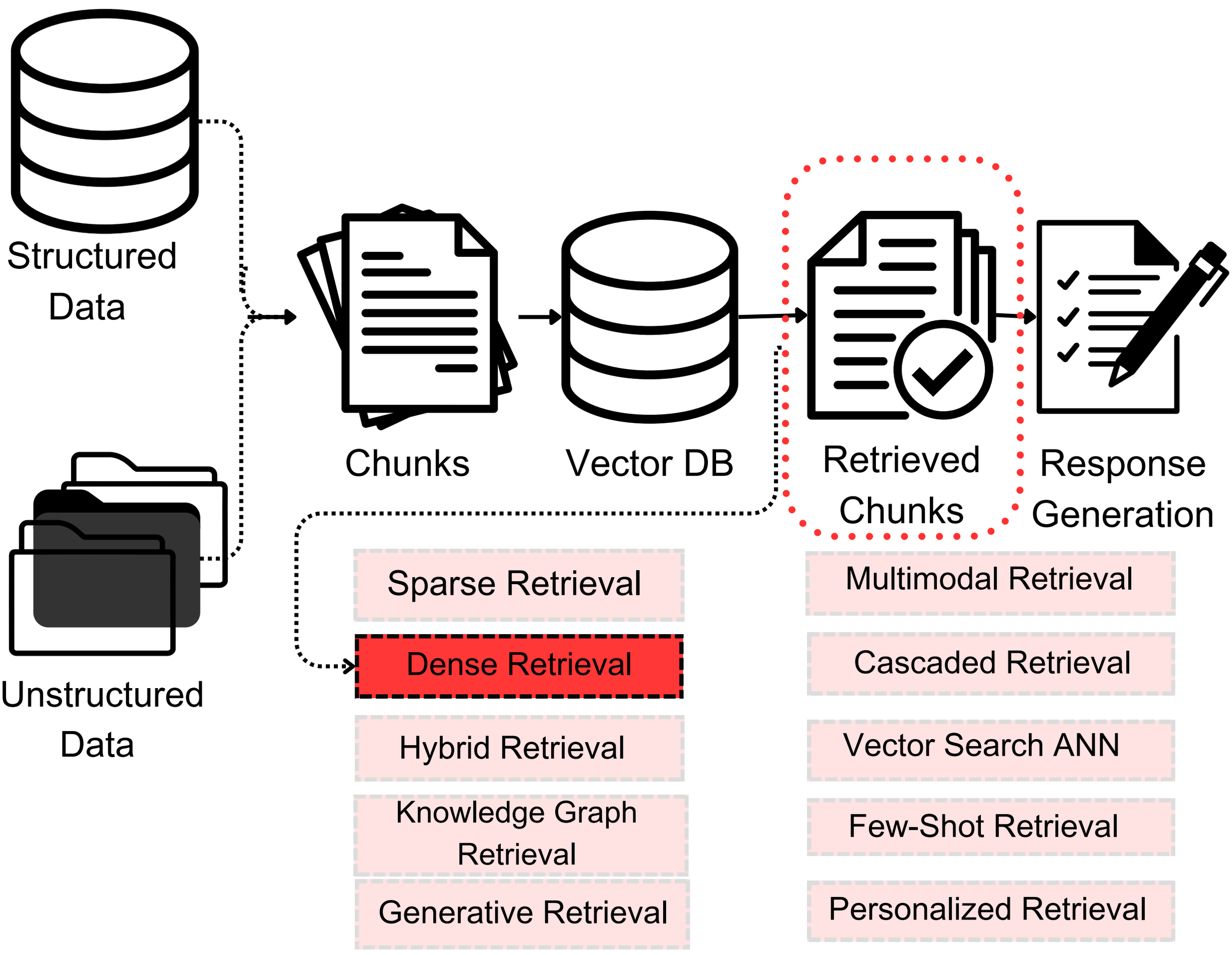
# Different Types of

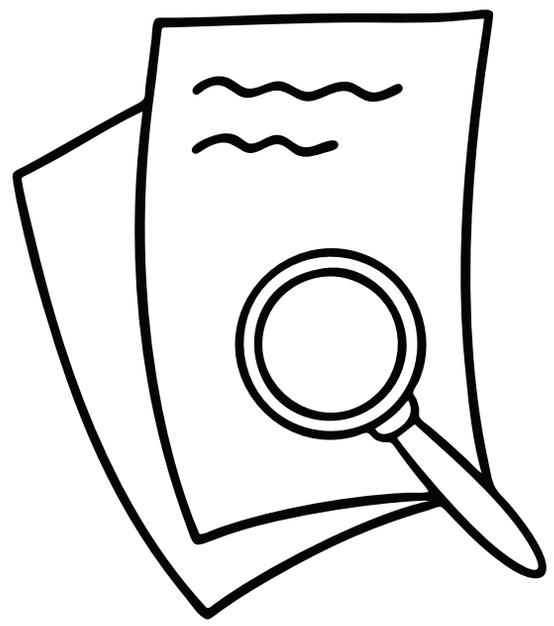
# Retrieval

# in RAG System



# Information

# Retrieval



Information Retrieval (IR) is the process of finding relevant information from large collections of unstructured data like text, documents, or multimedia.

## How it works?

1. A user query is submitted (e.g., a question or keywords).
2. The system searches through available data (e.g., documents or embeddings).
3. Results are ranked and displayed based on relevance.

## Why is it important?

1. Powers RAG-based systems .
2. Enables context-aware conversational AI.
3. Supports knowledge-intensive tasks.

## Key Components

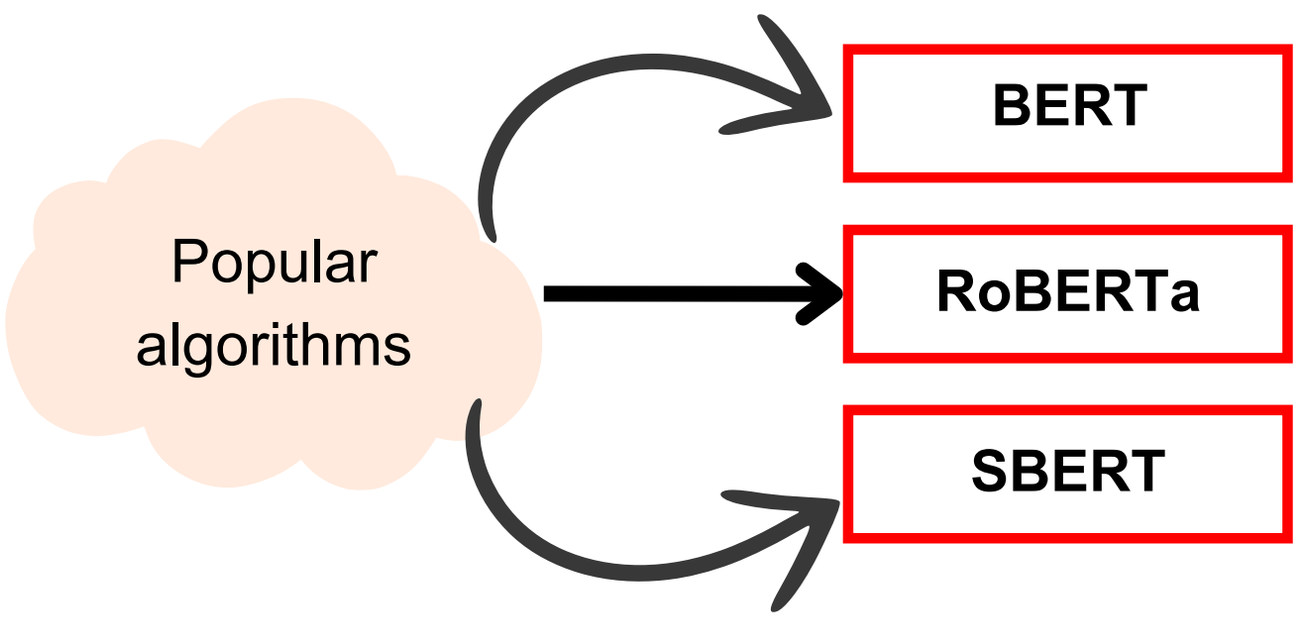
🔍 Retrieval Models: How the system retrieves data (e.g., Sparse, Dense).
📊 Ranking Algorithms: Ensuring the best results appear first.
🔄 Relevance Feedback: Continuous improvement based on user interaction.

# Dense

# Retrieval

🔍 Dense Retrieval is a technique in Information Retrieval (IR) where queries and documents are represented as dense vectors using deep learning models like BERT, RoBERTa, and SBERT.

Unlike traditional keyword-based search (Sparse Retrieval, e.g., BM25), dense retrieval captures semantic meaning, making search results more relevant and context-aware.




## How it works?

| | |
|---|---|
| Embedding Generation | 1 Convert queries and documents into vector representations using neural networks. |
| Vector Indexing | 2 Store embeddings in a vector database (e.g., FAISS, Annoy, ScaNN). |
| Similarity Search | 3 Use cosine similarity or dot product to retrieve the most relevant documents. |

# Dense

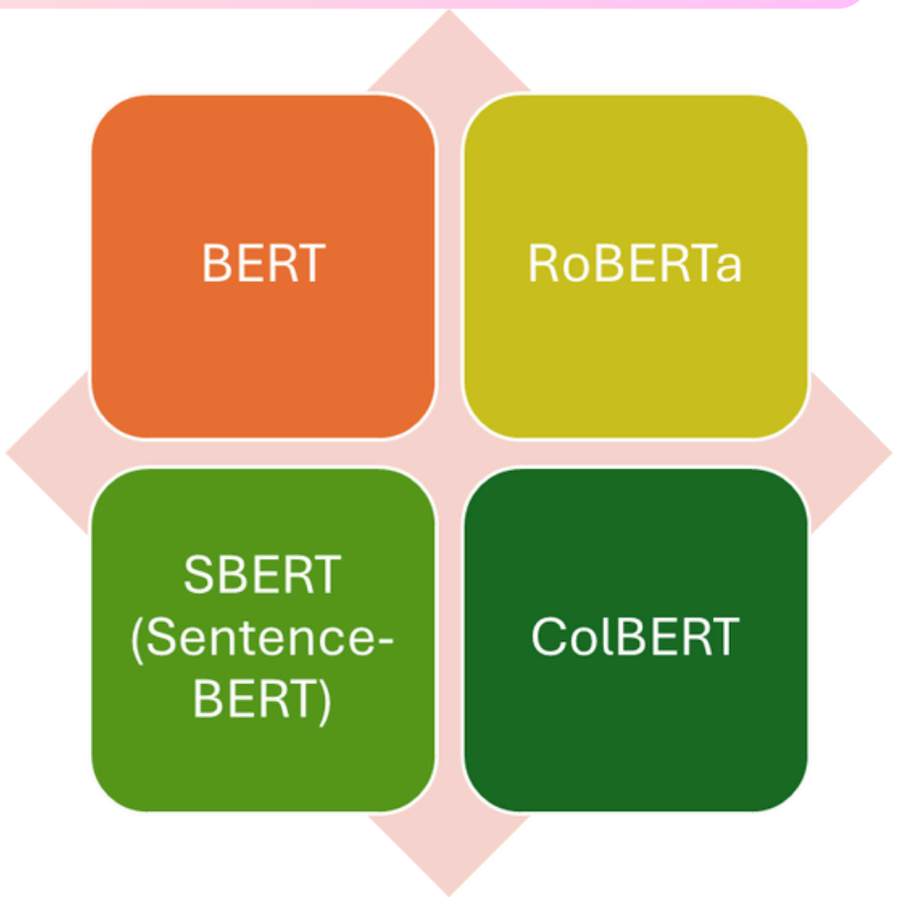
# Retrieval

| Feature | Sparse Retrieval (BM25) | Dense Retrieval (BERT-based) |
|---|---|---|
| Approach | Keyword Matching | Semantic Matching |
| Speed | Fast | Requires Optimization |
| Relevance | Lower (Lexical-based) | Higher (Context-aware) |
| Storage | Small | Large (Needs embeddings) |

## Why is Dense Retrieval Important?

✅ Handles Synonyms & Variations – “car” vs. “automobile”
✅ Understands Context – “Apple” (fruit) vs. “Apple” (company)
✅ Enhances Search in Large-Scale Datasets
✅ Boosts AI-powered Chatbots & Assistants

## Popular Dense Retrieval Models & Tools



# Dense Retrieval - Best Practices

## BERT

| | |
|---|---|
| Architecture | Transformer-based model. |
| Training Objective | Masked Language Model and Next Sentence Prediction |
| Use Case | General-purpose language understanding tasks. |

## BERT (Sentence-BERT)

| | |
|---|---|
| Architecture | BERT-based model fine-tuned for sentence embeddings. |
| Training Objective | Siamese network structure to produce semantically meaningful sentence embeddings. |
| Use Case | Sentence similarity, dense retrieval. |

## ColBERT (Contextualized Late Interaction over BERT)

| | |
|---|---|
| Architecture | BERT-based model with late interaction mechanism. |
| Training Objective | Optimized for efficient and effective retrieval. |
| Use Case | Information retrieval with efficient query-document interaction. |

# Dense Retrieval - Best Practices

Which is better?

- For **general language understanding tasks**, **RoBERTa** might be better due to its optimizations.
- For **sentence similarity and dense retrieval**, **SBERT** is usually better due to its fine-tuning for these tasks.
- **ColBERT** is optimized for efficient retrieval in **large-scale information retrieval systems.**

## Use Cases of Dense Retrieval

| | |
|---|---|
| Enterprise Search | Find relevant documents instantly |
| Question Answering (Q&A) | Used in ChatGPT, Google Search |
| E-commerce Search | Better product recommendations |
| Healthcare & Legal Search | Retrieve key insights from complex documents |

# Dense Retrieval

Challenges

- High computational cost
- Large storage for embeddings
- Latency issues

Optimizations

- ✅ Use Approximate Nearest Neighbor (ANN) search
- ✅ Dimensionality reduction for smaller embeddings
- ✅ Hybrid search (Dense + Sparse) for better recall

LLM-powered Embeddings

Models like OpenAI’s Ada, Mistral, and Llama3 generate high-quality dense embeddings, improving retrieval accuracy.

# Dense Retrieval in Python - Bert

Step1

Install the necessary libraries:

```
%pip install transformers torch
```

Step2

Load a pre-trained BERT model and tokenizer:

```
from transformers import BertTokenizer, BertModel
import torch

# Load pre-trained model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
```

✓ 32.6s

Step3

Encode the query and documents:

```
def encode(text, tokenizer, model):
    inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).squeeze()

query = "What is dense retrieval?"
documents = [
    "Dense retrieval uses dense vector representations.",
    "Sparse retrieval uses term-based representations.",
    "BERT is a transformer model."
]

query_vector = encode(query, tokenizer, model)
document_vectors = [encode(doc, tokenizer, model) for doc in documents]
```

# Dense Retrieval in Python - Bert

Step4

Compute similarity scores:

```python
from sklearn.metrics.pairwise import cosine_similarity

def get_similarity(query_vector, document_vectors):
    similarities = cosine_similarity([query_vector], document_vectors)
    return similarities[0]

similarities = get_similarity(query_vector, document_vectors)
```

Step5

Retrieve the most relevant document:

Generate Code

```python
most_relevant_doc_index = similarities.argmax()
most_relevant_doc = documents[most_relevant_doc_index]

print(f"Query: {query}")
print(f"Most relevant document: {most_relevant_doc}")
```

✓ 0.0s

```
Query: What is dense retrieval?
Most relevant document: Dense retrieval uses dense vector representations.
```

# CONGRATULATIONS

You have reached the end, now

If you want to help your network

REPOST THIS

Sarveshwaran R

https://github.com/DataSphereX/Retrieval-Strategies