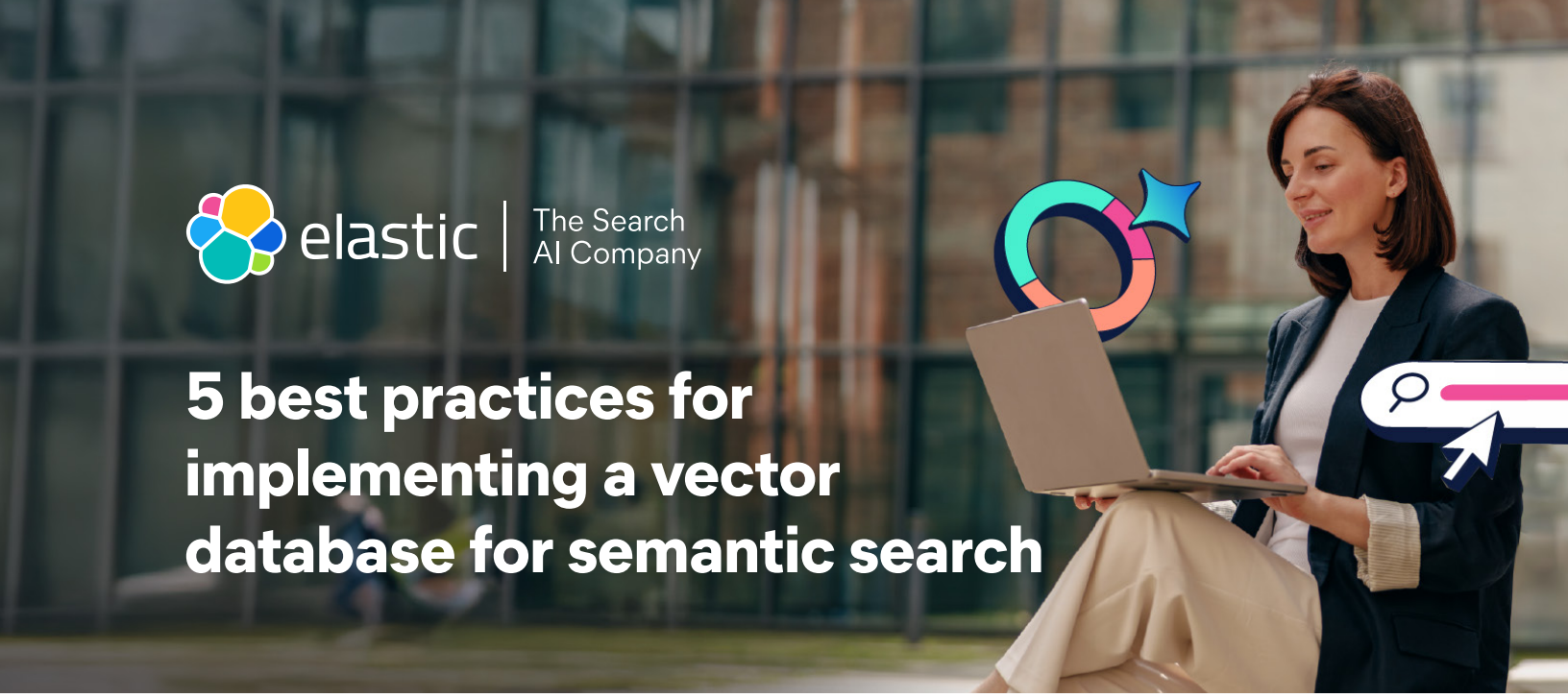

elastic | The Search AI Company

5 best practices for implementing a vector database for semantic search

A new era in information retrieval

As artificial intelligence (AI) becomes integral to modern applications, the way organizations approach search and information retrieval is fundamentally changing. Traditional keyword-based search is yielding to semantic search, interpreting the meaning and intent behind user queries.

Vector databases are at the core of this transformation. By storing and indexing data as vector embeddings, they enable high-precision, low-latency retrieval across massive volumes of unstructured and semi-structured data. This capability is essential for powering advanced AI use cases, like forms of in-context learning like retrieval augmented generation (RAG). Across industries, vector databases are unlocking new possibilities for knowledge discovery, automation, and user engagement. With built-in support for both dense and sparse vector models, and features that simplify data ingestion and indexing, platforms like Elasticsearch make it easier than ever for developers to implement scalable, high-performance semantic search.

This ebook explores the fundamentals and features of vector databases, plus five best practices for implementation. Whether you are building intelligent search, automating workflows, or enabling real-time AI experiences, understanding vector databases is essential for delivering the next generation of search and retrieval solutions.

Understanding vector databases

What is a vector database?

A vector database stores information as vectors — numerical representations of data objects known as vector embeddings. These embeddings act as unique fingerprints, capturing the underlying meaning or features of the data. By leveraging vector embeddings, vector databases can efficiently index and search across large volumes of unstructured and semi-structured data, such as text, images, and other multimedia.

Key characteristics of vector databases include:

High-dimensional data handling: Purpose-built to store, index, and query large volumes of high-dimensional data, such as text embeddings, image features, or sensor readings

Semantic similarity search: Efficiently retrieves items based on vector similarity, enabling context-aware, relevant results that go beyond exact keyword matching

Performance at scale: Delivers high precision and low latency, ensuring accurate results and minimal response times even as datasets and query loads grow

Horizontal scalability: Designed to accommodate expanding data volumes and increased query demands without sacrificing performance

Implementation and best practices

Implementing a vector database for semantic search involves several key considerations, from model selection to data preparation and indexing strategies. The following best practices can help ensure optimal performance and relevance in your AI-powered search applications.

1. Choose the right vector model

There are two primary types of vector models:

Dense vector models: Dense vectors are arrays of floating-point numbers that capture the semantic meaning of data. These models offer high precision and can be fine-tuned for specific datasets, delivering strong in-domain performance. Fine-tuning requires a dedicated data science team and labeled data, which may not be feasible for every organization.

Sparse vector models: Sparse vectors represent data as token-weight pairs, often derived from models like the Elastic Learned Sparse EncodeR (ELSER). ELSER is a retrieval model trained by Elastic that enables you to perform semantic search to retrieve more relevant search results. ELSER does not require fine-tuning, making it adaptable for a wide range of use cases out of the box. To learn more about sparse vector embeddings, check out this article.

Model recommendations:

Use the E5 dense vector model for multilingual semantic search or when working with non-English documents and queries.

Use the ELSER sparse vector model for English-language semantic search. ELSER is trained by Elastic and does not require additional fine-tuning, providing relevant results based on contextual meaning and user intent.

2. Prepare and convert your data

Data cleaning: Ensure that your data is clean, well-structured, and free of inconsistencies before indexing.

Vectorization: Convert your data into vector representations using the selected model. This step is critical for enabling efficient semantic search.

3. Start indexing automatically

Leverage the `semantic_text` field type: Starting with Elasticsearch 8.15, the `semantic_text` field type simplifies semantic search implementation. It automates chunking and vectorization, allowing you to index large documents or transcripts without manual preprocessing. Simply provide the full text, and Elasticsearch will handle chunking and embedding behind the scenes. Learn how to perform semantic search with `semantic_text` in this step-by-step tutorial.

Index management: Regularly update and manage your indices to maintain performance and relevance as your data evolves.

4. Tune and distribute your queries

Query tuning: Optimize your queries to reduce latency and improve result accuracy. Take advantage of Elasticsearch's built-in query capabilities for both dense and sparse vector fields. Looking for more on relevance tuning for semantic search? Check out this article.

Load balancing: Distribute query loads evenly across your cluster to prevent bottlenecks and ensure consistent performance.

5. Implement continuous monitoring and maintenance

Performance monitoring: Continuously monitor system performance, including indexing and query latency, to identify and address potential issues.

Regular maintenance: Schedule routine maintenance tasks, such as index refreshes and resource allocation reviews, to ensure ongoing reliability and efficiency.

By following these best practices and leveraging features like the `semantic_text` field type, developers can streamline the implementation of vector search, reduce operational complexity, and deliver high-quality, context-aware search experiences.

Practical implementations by industry

Vector databases extend beyond powering chat applications for RAG. They enhance search outcomes, enable advanced automations, and support agent-driven workflows across industries. By leveraging semantic retrieval and advanced indexing, vector databases serve as the backbone for AI-enabled knowledge bases and intelligent automation.

Ecommerce

Product recommendation and similarity search: Use vector embeddings to recommend products similar to those a user has viewed or purchased, enhancing cross-sell and upsell opportunities.

Semantic catalog search: Enable customers to search product catalogs using natural language queries, retrieving relevant results based on meaning rather than exact keywords. Learn how to leverage hybrid search to build a better product catalog in this step-by-step tutorial.

Public sector

Document and knowledge base search: Empower government employees to semantically search large repositories of policies, regulations, and case files, surfacing the most contextually relevant information.

Citizen service automation: Use hybrid and semantic search to match citizen queries with the most appropriate public resources, forms, or services, improving response accuracy and efficiency.

Financial services

Fraud detection and investigation: Use vector search to identify patterns and similarities in transaction data, supporting the detection of anomalous or fraudulent activities.

Customer support and knowledge retrieval: Enable semantic search across support tickets, regulatory documents, and internal knowledge bases to quickly surface relevant answers for both customers and employees. Learn how to create an AI-powered case deflection application using the Elastic AI Assistant.

Scalable AI search solutions for any industry

Vector databases are rapidly becoming a foundational technology for AI-powered search and retrieval. By enabling semantic similarity search across high-dimensional data, they allow organizations to unlock deeper insights and deliver more relevant, context-aware results. Whether supporting advanced document search in the public sector, powering fraud detection and personalized support in financial services, or enhancing product discovery and customer interaction in ecommerce, vector databases provide the scalability, performance, and flexibility required for modern applications.

Implementing vector search with Elasticsearch is streamlined by built-in support for both dense and sparse vector models, including E5 and ELSER. Features like the `semantic_text` field type further simplify data ingestion and indexing, automating complex tasks such as chunking and vectorization. By following best practices in model selection, data preparation, indexing, query optimization, and system maintenance, developers can build robust, efficient, and scalable AI search solutions. For a practical walkthrough of how to navigate an Elastic vector database, check out this article.

As the landscape of AI and search continues to evolve, vector databases will remain central to delivering intelligent, real-time experiences that meet the demands of users and organizations alike.

Visit our website to learn more about using Elasticsearch as a vector database.

