Vector Databases: A Beginner’s Guide

Prof. Dr.-Ing. Olaf Herden, Baden-Wuerttemberg Cooperative State University, Stuttgart - Campus Horb, Germany

Use cases such as LLMs (large language models), NLP (natural language processing) in interactive systems, semantic search in multimedia data and similarity search in recommender systems have increased the demand for efficiently storing and retrieving large vector data. Because traditional (relational) databases struggle to manage this type of data efficiently, there is a need to use vector databases. This tutorial introduces the core concepts of vector databases and provides an overview of common search, similarity and indexing techniques in the field.

Most data, including text, social media posts, images and videos, and audio, are unstructured. Storing this kind of data in relational databases is difficult to realize because these data cannot be searched directly. For comparing and searching unstructured data we must annotate it with keywords and categories. The search is based on these keywords.

Therefore, it is necessary to store unstructured data in a different format. The solution is to convert the data into vectors, arrays of real numbers. The vector representation of a piece of data is also called an embedding. Vector databases store these vectors and offer similarity search on them. To speed up the search, the vectors are typically indexed. The search task involves finding the most similar vectors in the database. These vectors are the nearest neighbors of the search pattern and therefore the algorithm is called NN algorithm (nearest neighbor). In the context of vector databases this search is often implemented with a degree of fuzziness by applying the ANN algorithm (approximate nearest neighbor).

As a basis for searching, we need to define the similarity between two vectors. Similarity can be quantified in terms of a distance in a metric space such as Euclidean distance or by cosine similarity.

Regardless of the similarity measure used, comparing vectors is a computationally expensive task. So, there is a need to index the vectors in the database to speed up searching. Various approaches exist, such as product quantization, locality sensitive hashing or the (hierarchical) navigable small world approach.