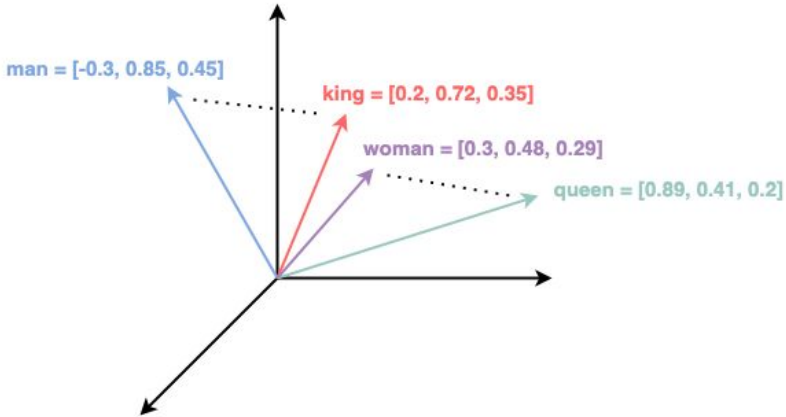
Understanding Vector Database

Introduction

- Semantic search and retrieval-augmented generation (RAG) applications require systems to be able to save lots of embedding vectors and also be able to retrieve the most relevant vectors with low latency. This requirement has resulted in emergence of kind of databases called vector databases.
- **Need for vector database** : Vector databases are designed to efficiently store and search for data based on its vector representation. This makes them ideal for applications that require similarity search, such as:
    - Product recommendation systems Image search
    - Natural language processing Fraud detection
    - Recommendation systems

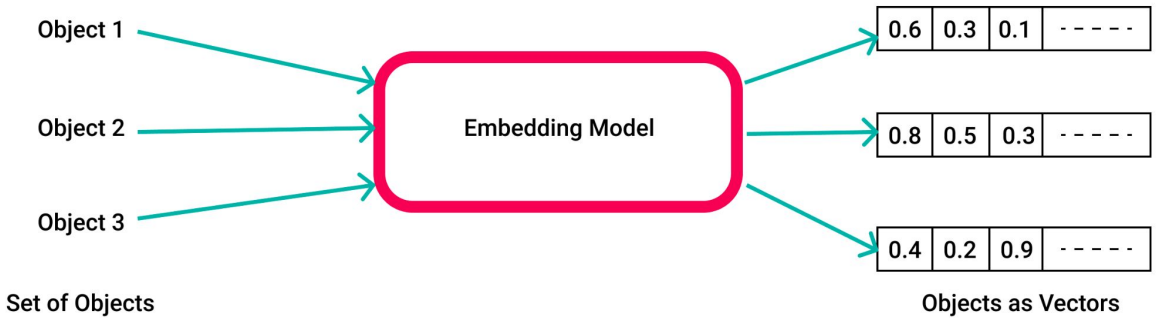
Vector Embedding

- An embedding vector is a form of representation that converts objects (like words, items, users, images, etc.) into vectors of real numbers in a lower-dimensional space.
- Vectors capture the essential and meaningful relationships between the objects .
- A basic example - Word Embeddings : These are used to map words or phrases from the vocabulary to vectors of real numbers. Examples of word embedding models include Word2Vec, GloVe.
- For instance, the embedding might capture relationships like "king" - "man" + "woman" ≈ "queen".



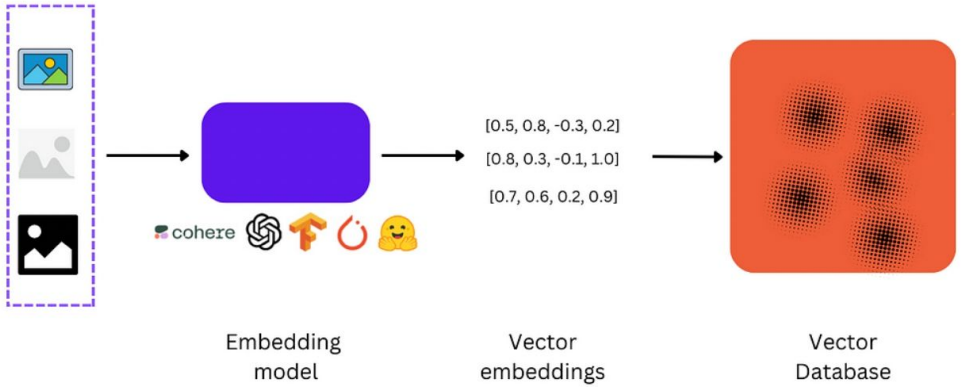
Embedding model

- Vector databases use embedding models as a key component for translating data into vector formats optimized for similarity search and pattern analysis.
- The embedding models produce the vector representations that vector databases are built to store, query and analyze.



Vector database

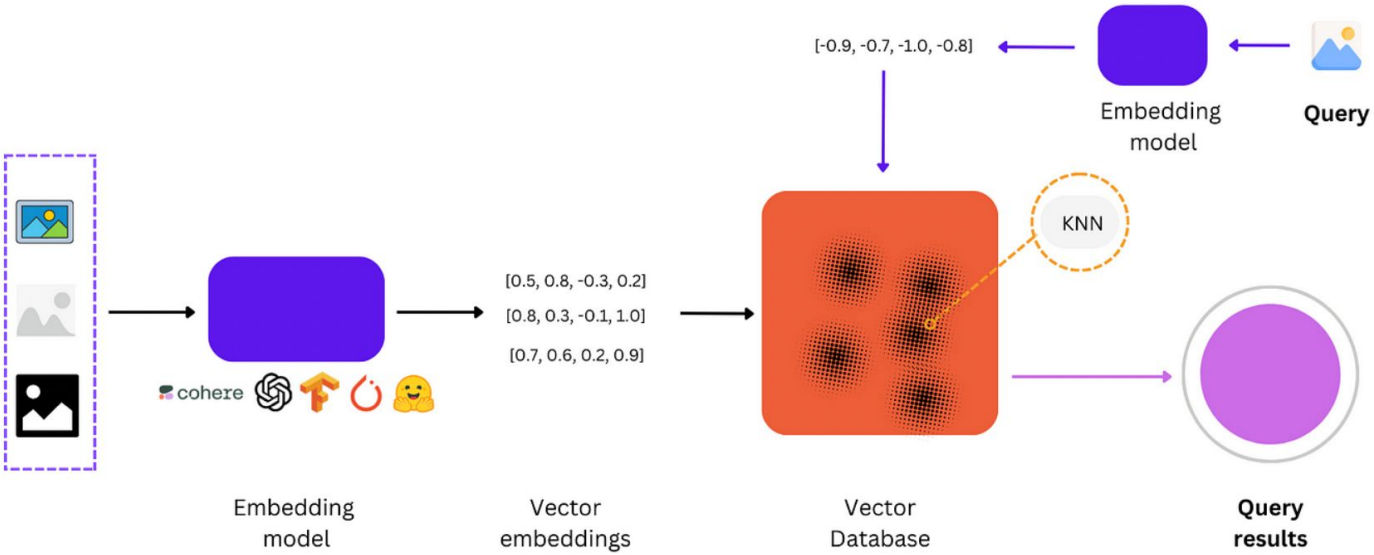
- A vector database is a type of database that stores data as high-dimensional vectors.
- Each vector represents a single entity, such as a text, image, or audio clip.
- The vectors are usually generated by applying some kind of transformation or embedding function to the raw data.
- This function can be based on various methods, such as ML Embedding models, word embeddings, or feature extraction algorithms.



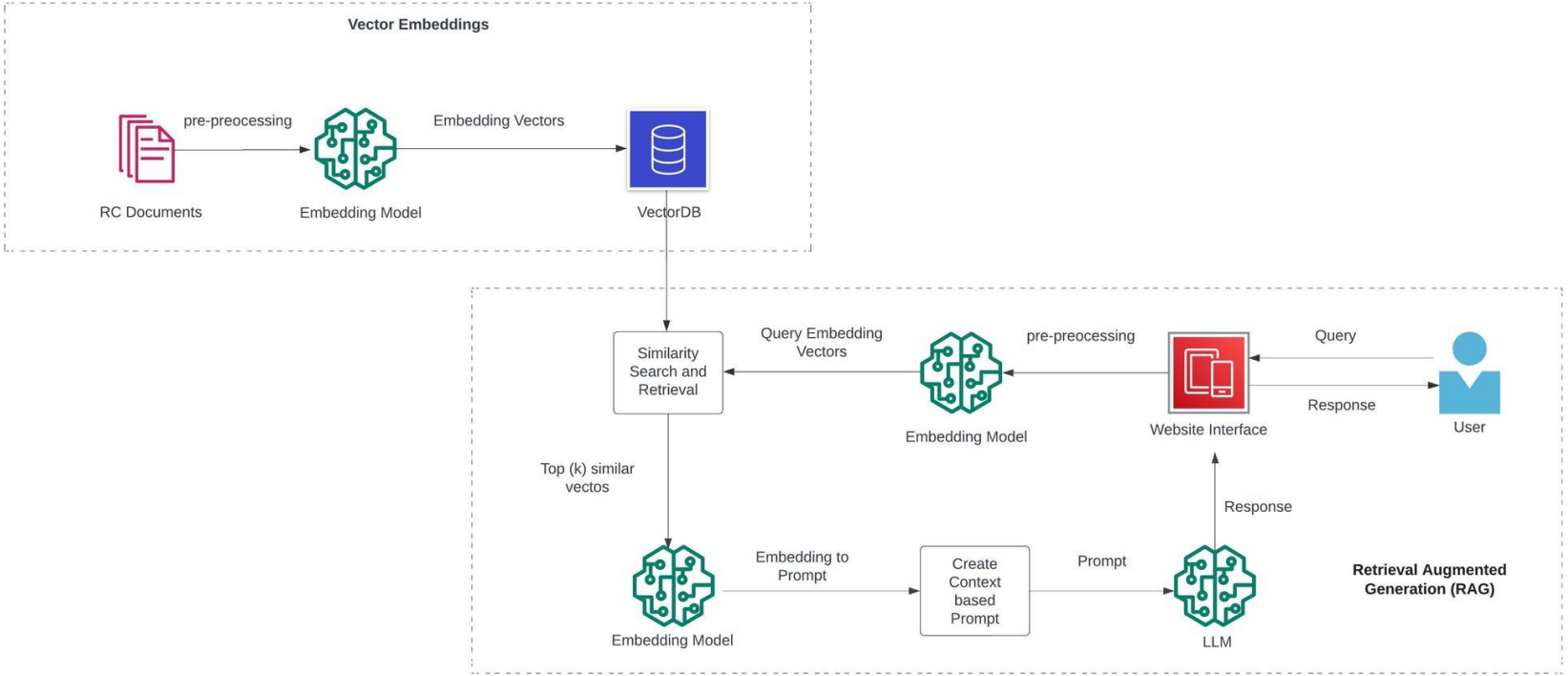
Comparing Vector DBs to other DBs:

Attribute	Vector Database	Relational Database	NoSQL Database	Graph Database
Data Structure	Vectors	Tables	Key-value pairs, Documents, Wide-column, Graph	Graph
Data Model	Vector Space Model	Relational Model	Various models depending on the type	Graph Model
Query Language	Similarity-based queries	SQL	Specific to the database	Graph query languages (e.g., Cypher for Neo4j)
Schema Type	Schema-less	Schema-based	Schema-less or flexible schema	Schema-less or flexible schema
Data Relationships	Not available	Defined by relations/foreign keys	Not available	Modeled using edges and nodes
Use Cases	Similarity search, Recommendation systems, Machine learning	Traditional business applications, Transactional systems	Big data, Real-time applications, Unstructured data	Social networks, Fraud detection, Recommendation engines, Knowledge graphs

ML pipeline with vector database used to store embeddings



Retrieval Augmented Generation (RAG)



Comparing Vector DBs:

Common evaluation criteria include :

- Managed vs Self-Hosted
- Performance
- Existing MLOps
- Developer experience
- Reliability
- Security
- Cost

Comparing Vector DBs:

Attribute	Chroma	Milvus	Pinecone	Qdrant
Type	Open Source	Open Source & Paid	Commercial	Open Source
Description	High performance and scalability, based on Faiss library	Real-time search and recommendations, highly scalable, based on HNSW algorithm	Designed for enterprise applications, providing high performance, scalability, security, and compliance	New but balanced between performance and simplicity, high performance
Key Features	Support variety of vector data, Distributed indexing, Real-time query processing	High performance, Scalability, Various vector types support, Open source, Python SDK, Takes resources locally	High performance, Scalability, Security and compliance, Integration with various data sources, All major distance metrics	Open source, Excellent documentation, Intuitive API, High performance, New with smaller community
Pricing	Free	Free and Paid, - Capacity-optimized on Zilliz cloud: $450/month - Cost optimized: $300/month - Performance optimized: $1,375/month	- Starter: Free (Limited to one index and one project) - Standard: $70/month (Est. for one index on one s1 pod for 30 days at $0.096/hour) - Enterprise: $104/month (Est. for one index on one s1 pod for 30 days at $0.144/hour)	Self-hosted: Free, - Storage optimized: $280/month - Not storage optimized: $820/month
QPS (Queries per second)	Data not available	1,751	Up to 200	300
Hosting	Self-hosted	Self-hosted and Cloud managed	Managed in Cloud	Self-hosted and Cloud managed