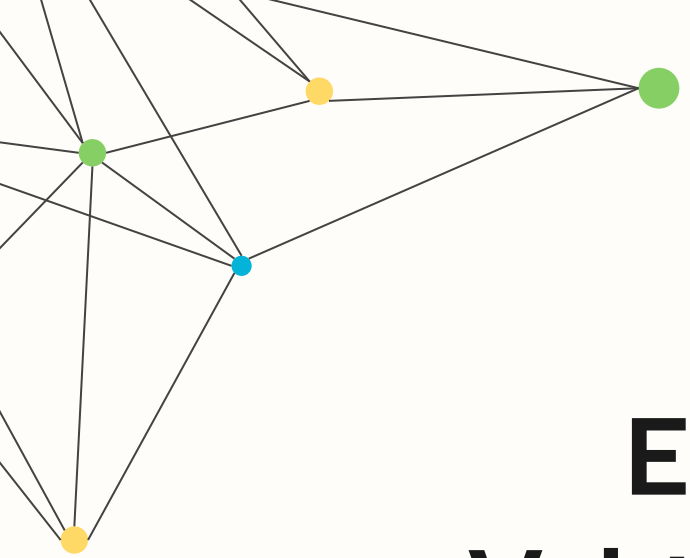
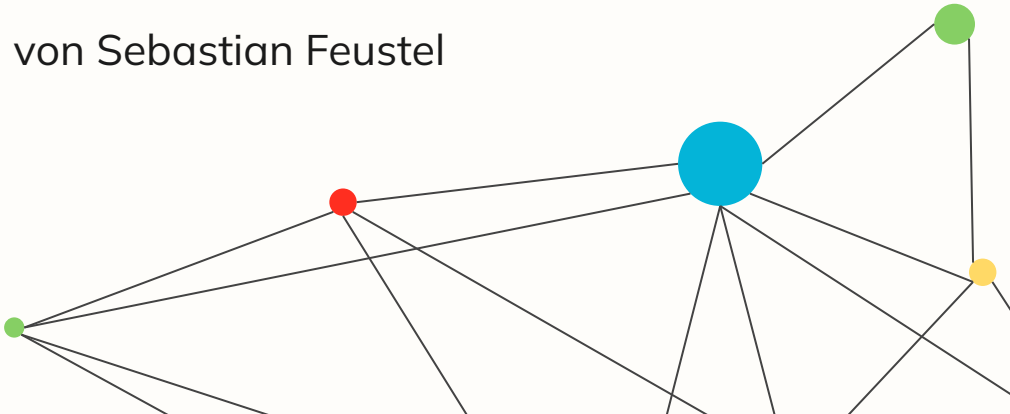
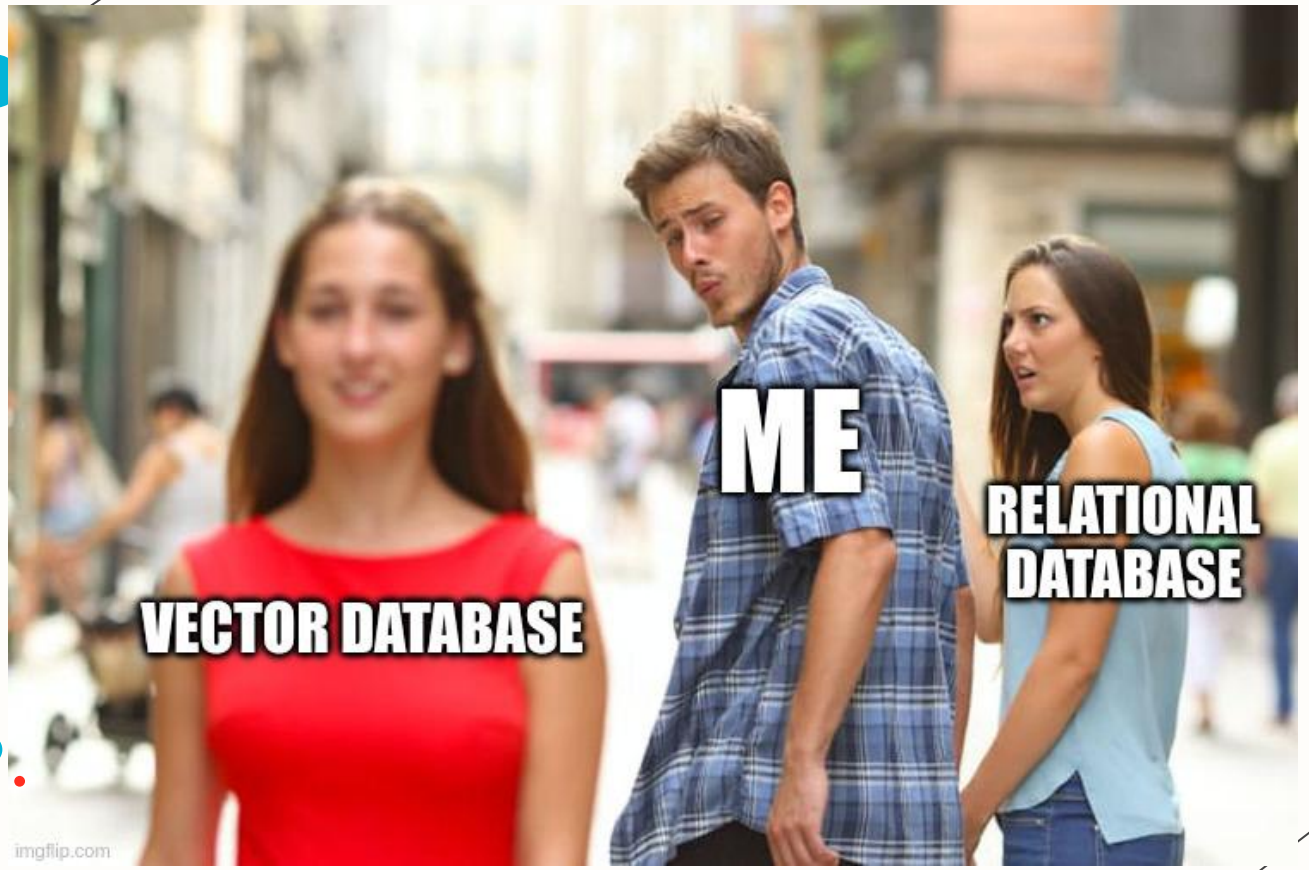


# Einführung in Vektordatenbanken

Eine Präsentation von Sebastian Feustel




VECTOR DATABASE
ME
RELATIONAL DATABASE
imgflip.com

# Table of contents


01 Theoretische Grundlagen

02 Arbeitsblatt

03 Mehr Theorie

04 Meine Projekte

# 01

# Theoretische Grundlagen

# Was sind Vektoren?

Ein **Vektor** ist im Kern eine geordnete Liste von Zahlen, die eine Größe beschreibt.

Ein einfacher Vektor sieht z.B. so aus:

$\vec{v}=(2,3)$ oder
im 3D-Raum: $\vec{v}=(2,3,5)$

Das bedeutet:

- Jeder Eintrag ist eine **Komponente**
- Der Vektor beschreibt einen **Punkt oder eine Richtung im Raum**
- In 2D oder 3D kann man sich das geometrisch gut vorstellen (Pfeil im Raum)

Ein **hochdimensionaler Vektor** ist genau das gleiche Prinzip nur mit sehr vielen Komponenten:

$\vec{x}=(x1,x2,x3,\ldots,x768)$

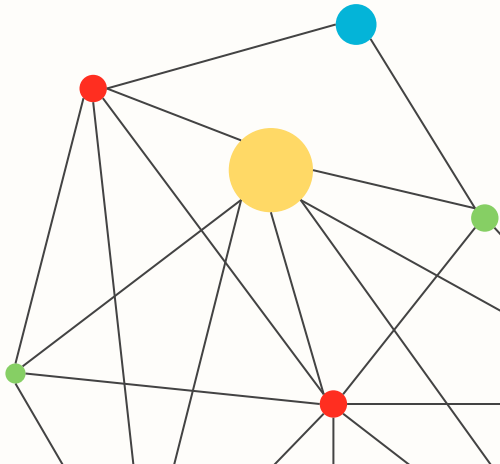
Typisch:

- 128, 384, 768, 1536 Dimensionen (z. B. bei Word Embeddings oder LLMs)
- Jeder Eintrag ist ein Feature oder eine gelernte Eigenschaft

# Grundlagen Vektordatenbanken

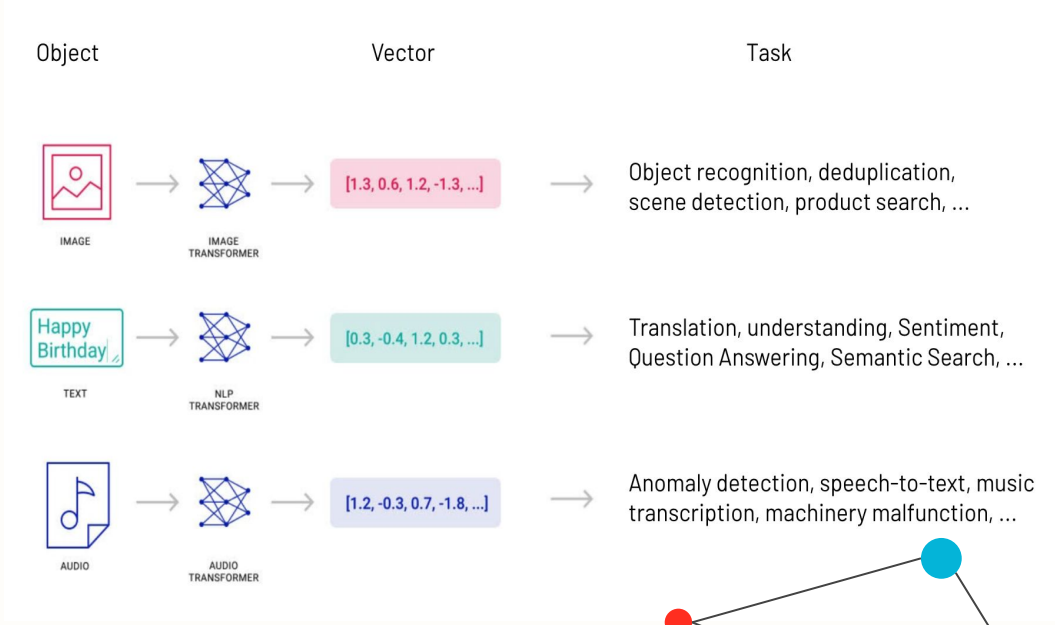
Eine **Vektordatenbank** ist speziell auf das Arbeiten mit Vektoren (meist aus KI/ML-Embeddings) ausgelegt.

- **Speicherung hochdimensionaler Vektoren**: Daten (z. B. Texte, Bilder, Audio) werden in numerische Vektoren transformiert und in der Datenbank abgelegt.
- **Ähnlichkeitssuche statt exakter Schlüsselabfrage**: Anfragen erfolgen über *Nearest Neighbor Search*, um ähnliche Objekte anhand von Distanz Maßen zu finden.
- **Optimierte Indexstrukturen**: Nutzt spezielle Datenstrukturen für schnelle Approximate Nearest Neighbor (ANN) Suchen.
- **Integration mit Machine Learning**: Oft nahtlos eingebunden in KI-Workflows (z. B. für RAG, Empfehlungs Systeme, Bilderkennung).
- **Skalierbarkeit und Verteilung**: Kann für große Datenmengen (Millionen bis Milliarden Vektoren) und häufig verteilt oder Cloud-native implementiert.



# Vektor-Embeddings

- Kompakte Repräsentation von Daten in der Form eines hochdimensionalen Vektors
- Embedding ist das Vorgehen (NLP) Text in Vektoren umwandeln und die Bedeutung und Beziehung zu erfassen
- ML-Modelle werden darauf trainiert, Daten zu Vector-Embeddings zu konvertieren



# Wie kommt man von Text auf Vektor?

Einfachste Methode: Bag of Words

Beispiel: “Als Anna abends aß, aß Anna abends Ananas.”

als -> 1

anna -> 2

abends -> 2

aß -> 2

ananas -> 1

Vektor: (1, 2, 2, 2, 1)

# Probleme mit Bag of Words

- Bag of Words hat keinen Kontext
- “Bank” (Sitzbank vs Geldinstitut) -> gleiche Representation
- Wortreihenfolge egal
- Semantik fehlt komplett

# Word Embeddings (erste Verbesserung)

## Beispiel: Word2Vec

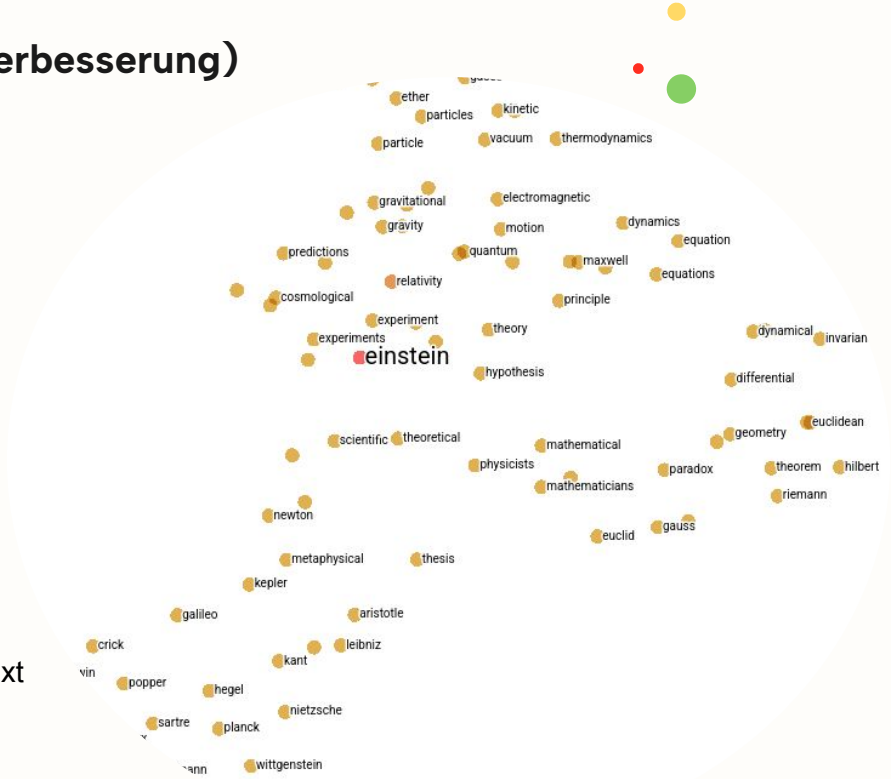
- Wörter werden als dichte Vektoren gelernt
- ähnliche Wörter liegen nah beieinander
- Beispiel:

  König – Mann + Frau ≈ Königin

- basiert auf **Kontextfenster**, aber:

  ❌ kein Satzverständnis

  ❌ gleiche Bedeutung unabhängig vom Kontext




https://projector.tensorflow.org

# Kontextuelle Embeddings (State of the Art)

**Beispiel: BERT**

- Wort bekommt **je nach Satz einen anderen Vektor**
- basiert auf Transformer-Architektur
- nutzt Attention (Kontext aus gesamtem Satz)

Beispiel:

- „Ich sitze auf der Bank“
- „Ich gehe zur Bank“

„Bank“ hat **zwei verschiedene Vektoren**

Vorteile:

- echtes Sprachverständnis
- besser für Suche / Chatbots

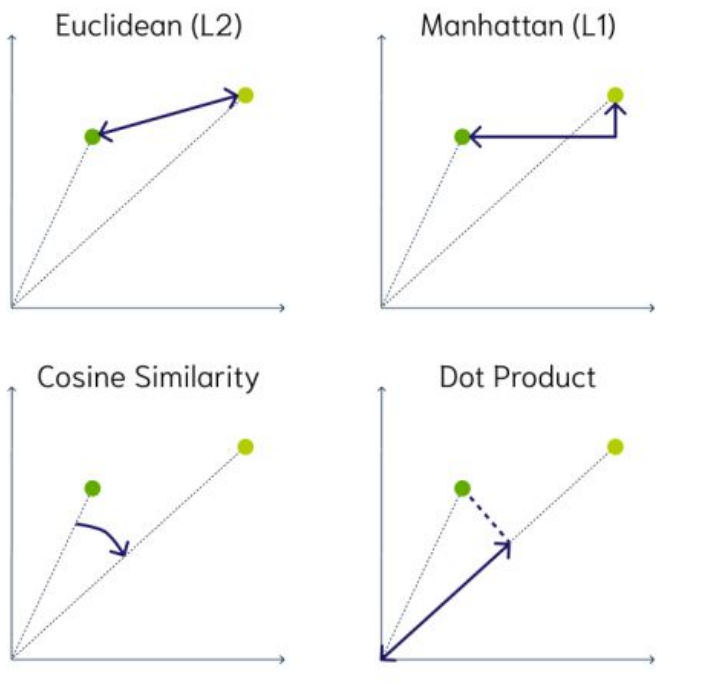
# Suchen/ Query

- Vektoren Queries suchen nach „gleichen“ Vektoren

Mehrere Metriken, um den Abstand zu messen:

- Euklidischer Abstand
- Manhattan
- **Kosinus Ähnlichkeit**

$$Kosinus - Ähnlichkeit = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\overrightarrow{||A||} \cdot \overrightarrow{||B||}}$$



# 02

# Arbeitsblatt

# Arbeitsblatt

Dieses Arbeitsblatt vermittelt die Grundlagen von Vektordatenbanken. Du lernst:

- Texte in Vektoren umzuwandeln (Bag of Words)
- Ähnlichkeiten zwischen Texten zu berechnen (Cosinus-Ähnlichkeit)
- Warum das für Suche und KI wichtig ist

Allgemeines Verfahren zur Berechnung des Skalarprodukt

$$\vec{a} \cdot \vec{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3$$

## Aufgabe 1: Bag-of-Words-Tabelle

| Wort | Satz 1 | Satz 2 | Satz 3 |
|---|---|---|---|
| der | 1 | 0 | 1 |
| die | 0 | 1 | 0 |
| hund | 1 | 0 | 1 |
| katze | 0 | 1 | 0 |
| spielt | 1 | 1 | 0 |
| schläft | 0 | 0 | 1 |
| im | 1 | 1 | 1 |
| garten | 1 | 0 | 0 |
| haus | 0 | 1 | 1 |

## Aufgabe 2: Vektoren

Reihenfolge wie oben:

- Satz 1: (1, 0, 1, 0, 1, 0, 1, 1, 0)
- Satz 2: (0, 1, 0, 1, 1, 0, 1, 0, 1)
- Satz 3: (1, 0, 1, 0, 0, 1, 1, 0, 1)

## Aufgabe 3: Skalarprodukt

- Satz 1 · Satz 2:

$$= (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) = 2$$

- Satz 1 · Satz 3:

$$= (1 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) + (1 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) = 3$$

## Aufgabe 4: Vektorlängen

- $\|\text{Satz 1}\| = \sqrt{5}$
- $\|\text{Satz 2}\| = \sqrt{5}$
- $\|\text{Satz 3}\| = \sqrt{5}$

## Aufgabe 5: Cosinus-Ähnlichkeit

- sim(Satz 1, Satz 2):

$$\frac{2}{\sqrt{5} \cdot \sqrt{5}} = \frac{2}{5} = 0{,}4$$

- sim(Satz 1, Satz 3):

$$\frac{3}{5} = 0{,}6$$

**Antwort:** Satz 3 ist ähnlicher zu Satz 1.

## Aufgabe 6

„Hund“ ist informativer, weil es ein inhaltstragendes Wort ist, während „der“ ein häufiges Funktionswort ist.

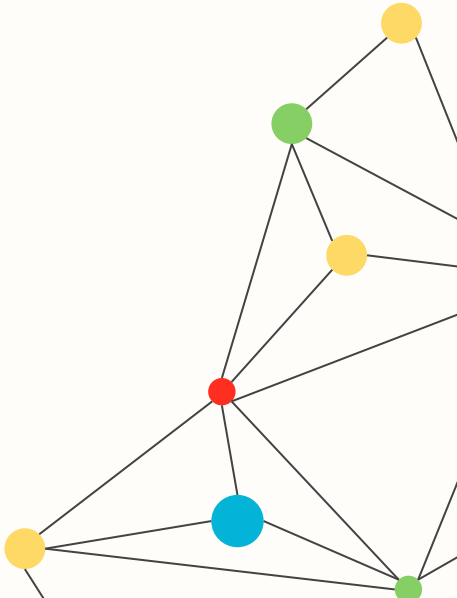
## Aufgabe 7

Das Wort erscheint nicht im Vektor (alle Werte 0), es trägt nichts zur Ähnlichkeit bei.

## Aufgabe 8

Bag of Words berücksichtigt keine Wortreihenfolge und keinen Kontext.

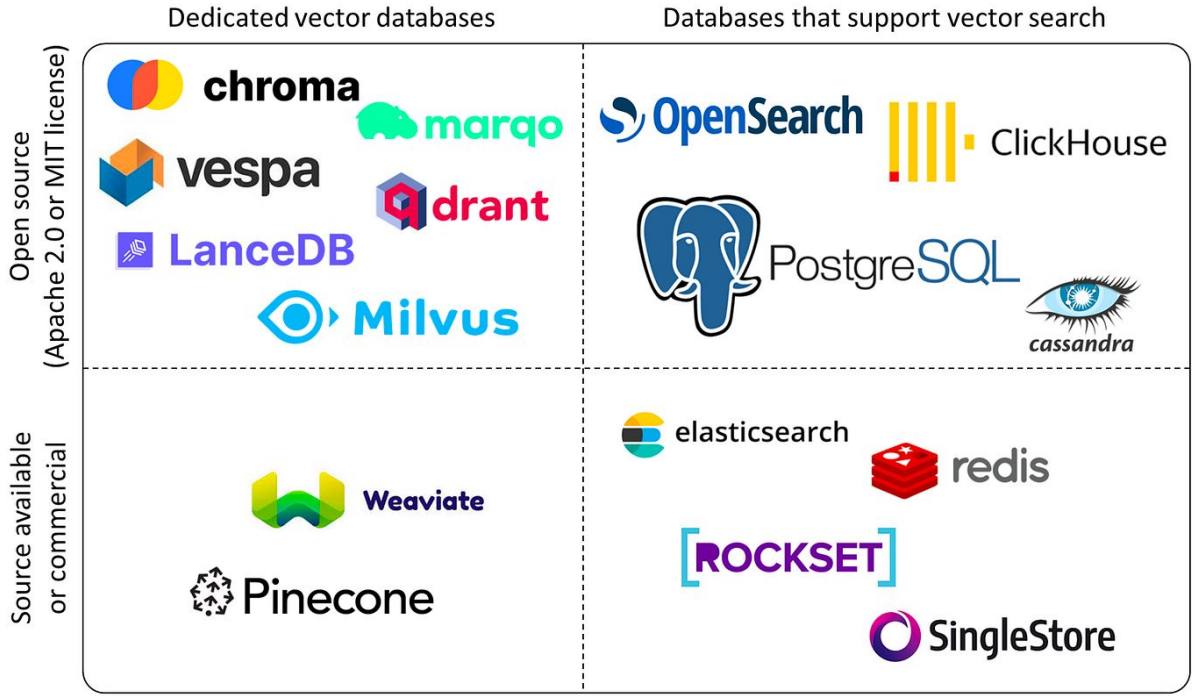
# 03
# Mehr Theorie



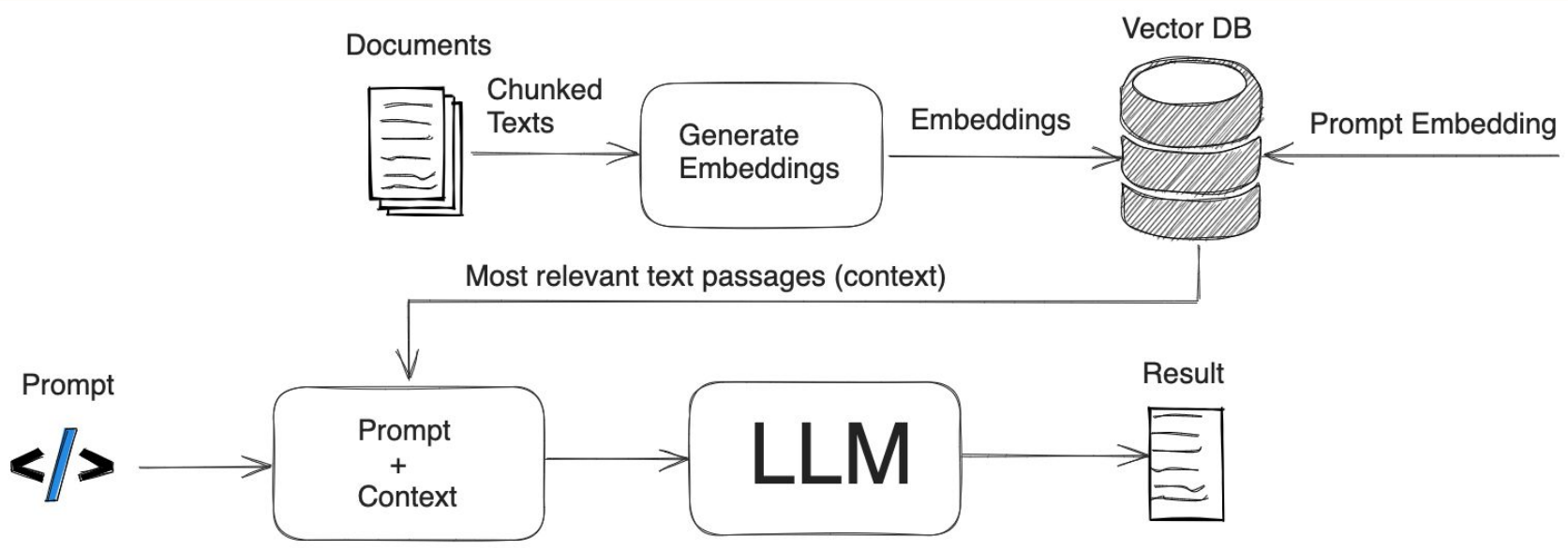
# INDEXING

Das sind **Methoden (Mittels/Verfahren)**, um in **Vektordatenbanken** oder allgemein in **Approximate Nearest Neighbor (ANN) Search** große Mengen an Vektoren effizient zu durchsuchen.

- **Hashing (LSH)** → schnell, weniger genau.
- **Graph (HNSW)** → genau & schnell, braucht aber viel Speicher.
- **Quantisierung (PQ)** → speichereffizient, aber ungenauer.


Dedicated vector databases
Databases that support vector search
Open source
(Apache 2.0 or MIT license)
Source available
or commercial
chroma
marqo
vespa
qdrant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
Weaviate
Pinecone
elasticsearch
redis
ROCKSET
SingleStore

# RAG（Retrieval Augmented Generation）



# Text Embedding Models

ollama.com/search?c=embedding

huggingface.co/spaces/mteb/leaderboard

# Bilder Embedding Models

https://github.com/openai/CLIP


https://github.com/facebookresearch/dinov3

# Audio Embedding Models

Empholen Sprache zu Text und Text dann zu Vektor

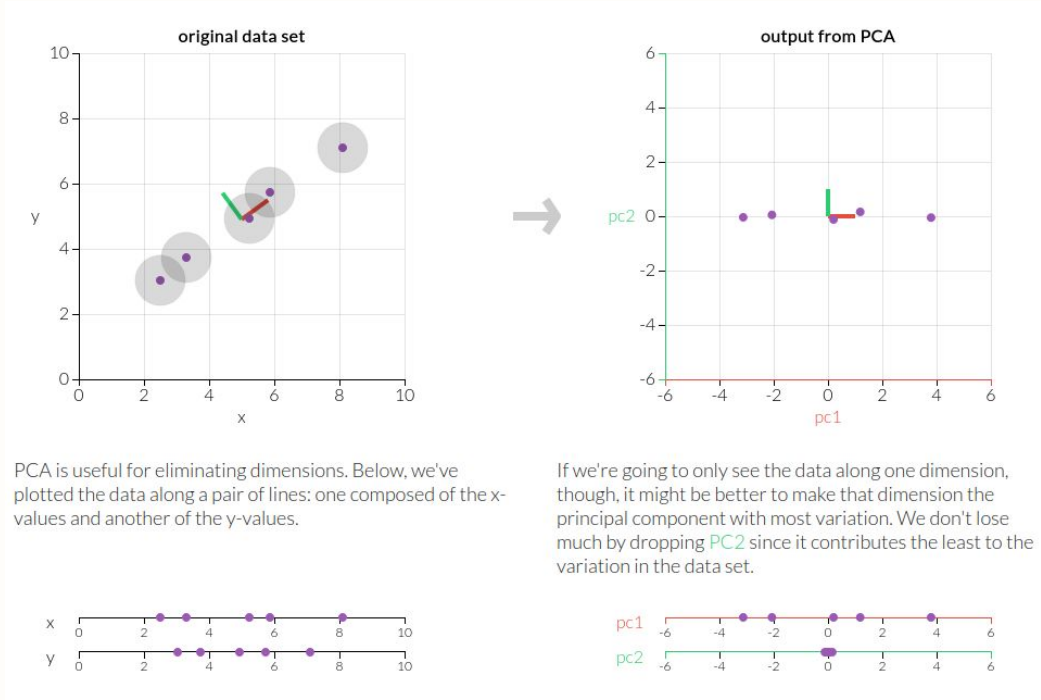
OpenL3 für Geräusche

BEATs (Microsoft)

CLAP (https://huggingface.co/laion/larger_clap_general)

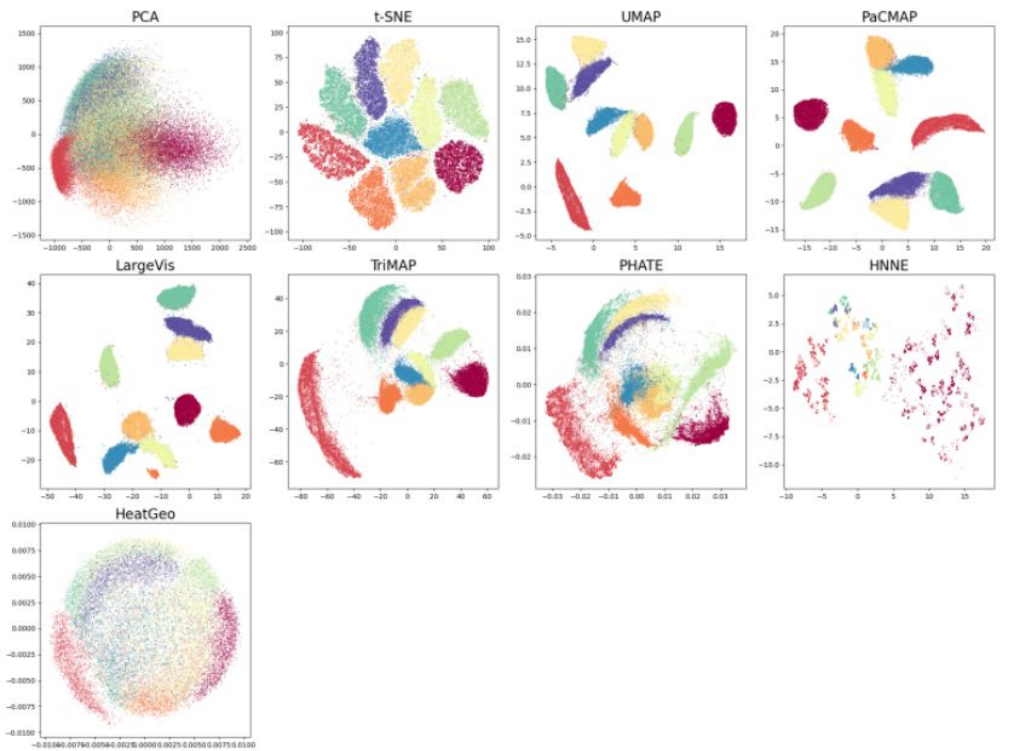
# Dimensionsreduktions-Algorithmus

# Dimensionsreduktions-Algorithmus




Quelle: https://setosa.io/ev/principal-component-analysis/

# Dimensionsreduktions-Algorithmus



# 04

# Eigne Projekte

# Dinofind

https://www.dinofind.com
https://github.com/53845714nF/dinofind

# Dinofind

**Kaggle**

30,000 Flickr images serve as the starting point for the image search.

**DINOv2**

Image Vectorizer converts images into high-dimensional embeddings using Vision Transformer.

**Qdrant**

Vector Database stores the embeddings for fast similarity searches

**MinIO**

Object Storage stores the original images in an S3-compatible bucket.

**Flask**

Serves as an HTTP server for searching and displaying results.

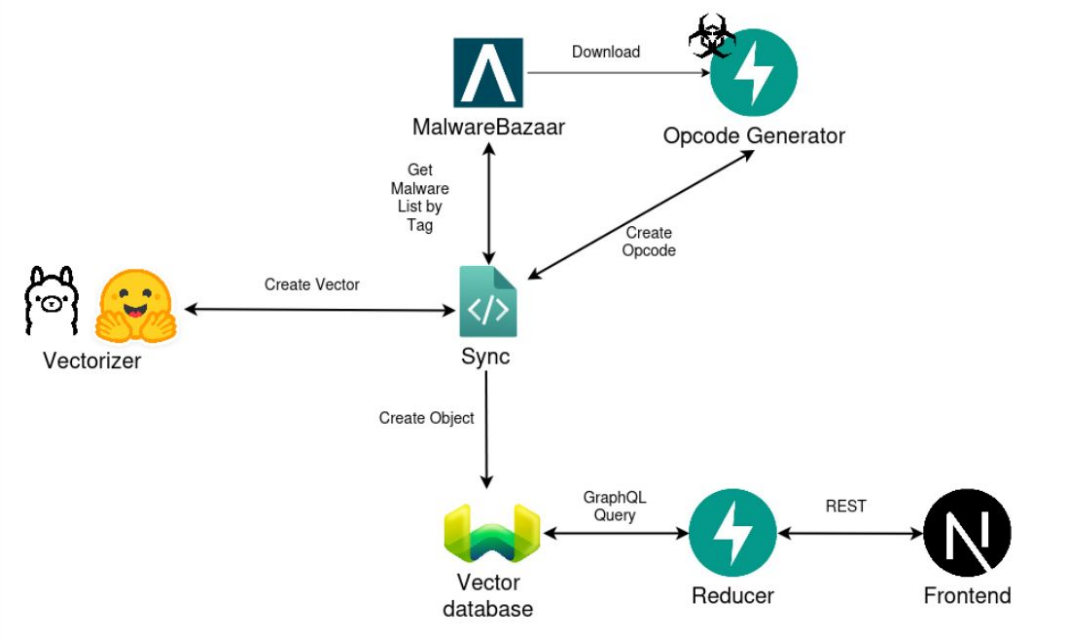
**Tailwind CSS**

Ensures a modern design of the user interface.

# Malwareuniverse

https://www.malwareuniverse.org/
https://github.com/malwareuniverse

# Malwareuniverse

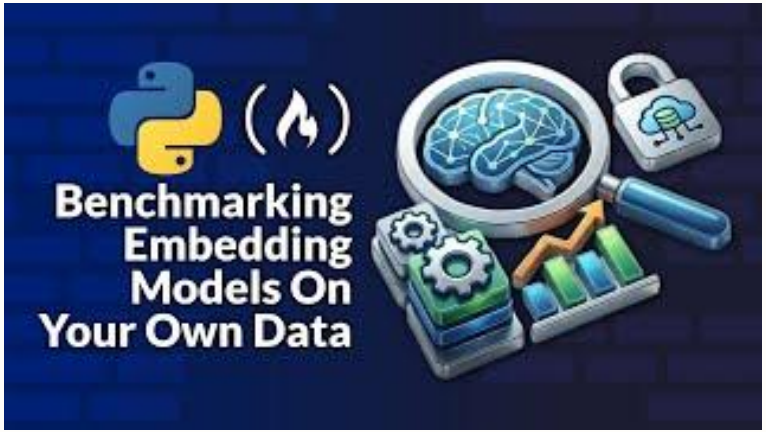


# 05

# Ausblick

# Ausblick

- **Benchmark Embedding Models**
- **Fine tuning/ PreTraining**
- **Vektoren Zusammenführen**



# Danke für Ihre Aufmerksamkeit!

## Haben Sie noch Fragen?

blog@hackwiki.de
www.sebastian-feustel.de