Vector Databases

A Technical Primer

Jide Ogunjobi

December 2023

TGE DATA LABS

tgedatalabs.com

Welcome to the Course

Hi, I'm Jide! 👋

A little about me:

- I like to describe myself as an end-to-end data professional (data analyst, data modeler, data architect, data engineer)
- I've spent the last few years building out data infrastructure and platforms at some great companies
- I've spent the better part of 2023 exploring vector databases and building applications that depend on Vector databases and have consulted with multiple companies regarding their AI based applications
- I also run a startup which helps startups and enterprise companies connect to and work with multiple vector databases
- Follow me on Linkedin

Course Structure

- **Section 1:** Introduction to Vector Databases
- **Section 2:** Vector Database Core Concepts
- **Section 3:** Understanding Search Similarity
- **Section 4:** Indexing and Querying
- **Section 5:** Working with Vector Databases
- **Section 6:** The Future of Vector Databases
- **Section 7**: Build a sample application

Introduction to Vector Databases

TGE
DATA LABS

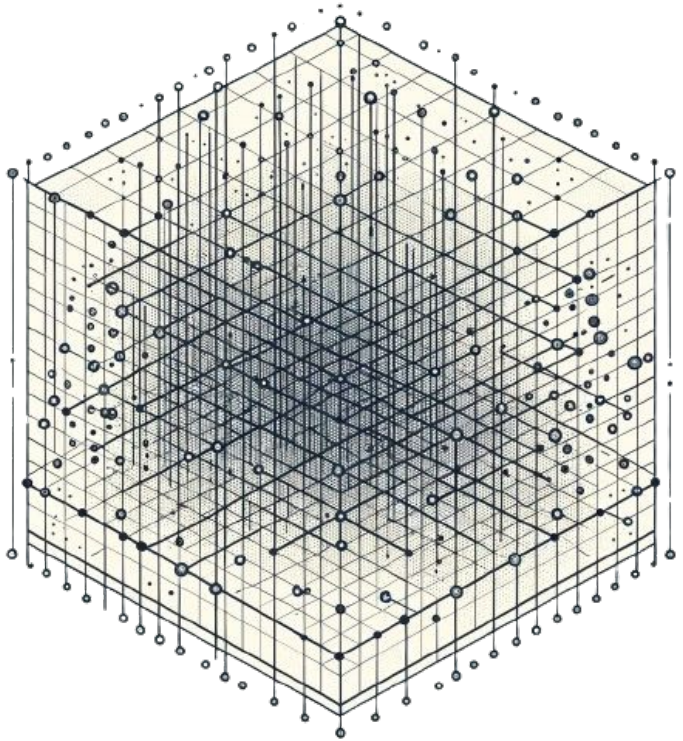
tgedatalabs.com

What are Vector Databases

Vector databases are a specialized type of database designed to store, manage, and process high-dimensional data representations known as vectors.

Unlike traditional databases that store data in rows and columns, vector databases store data as vectors in a multi-dimensional space.

Each vector represents mathematical arrays of numbers that represent the characteristics or attributes of data points.

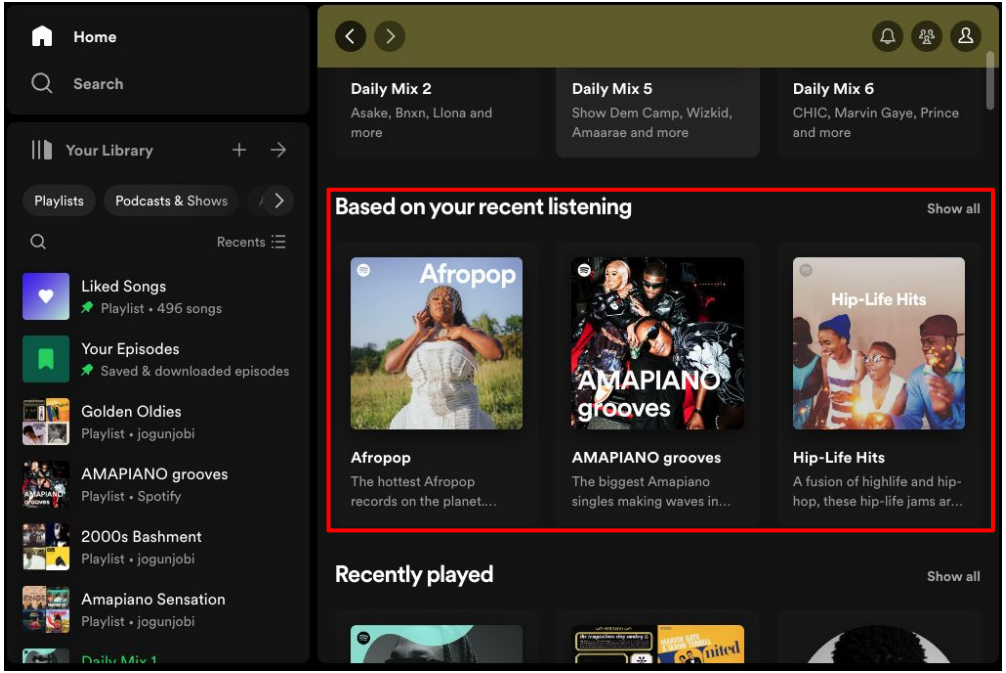


Key Principles of Vector Databases

Data Structured by Meaning

Unlike traditional databases that establish relationships between elements based on explicit links or hierarchy, vector databases associate records algorithmically based on similarity of their data attributes.

This enables intuitive connections to be established based on the implicit meaning within database elements rather than rigid schemas.



Key Principles of Vector Databases

Optimized for Analytic Queries

By encoding data as multi-dimensional vectors (i.e. number arrays) reflecting semantic relationships, vector databases can perform advanced analytic operations such as similarity search, clustering, and classification far faster than conventional systems.

Their computational model is ideal for pattern detection, predictive analytics, and other applications with demanding analytic requirements.

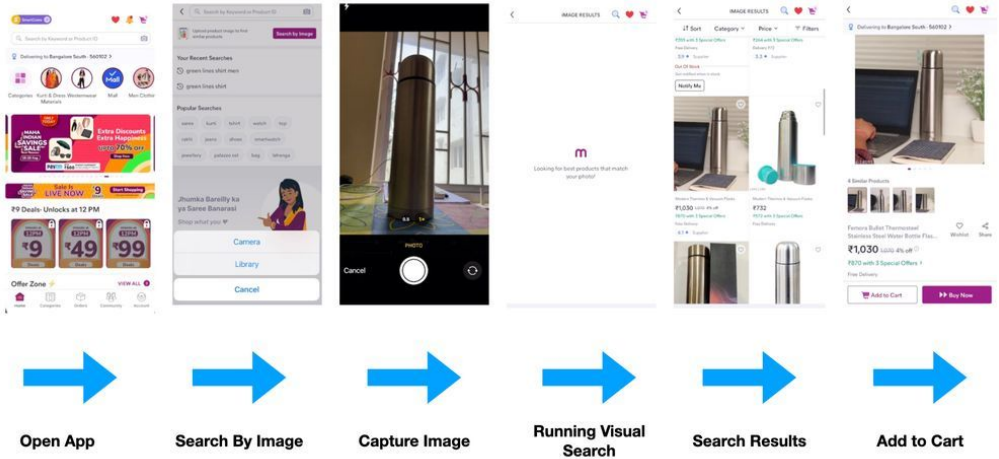


Image courtesy of meesho.io

Why are Vector Databases all the rage?

Emergence of AI and Machine Learning

TGE
DATA LABS

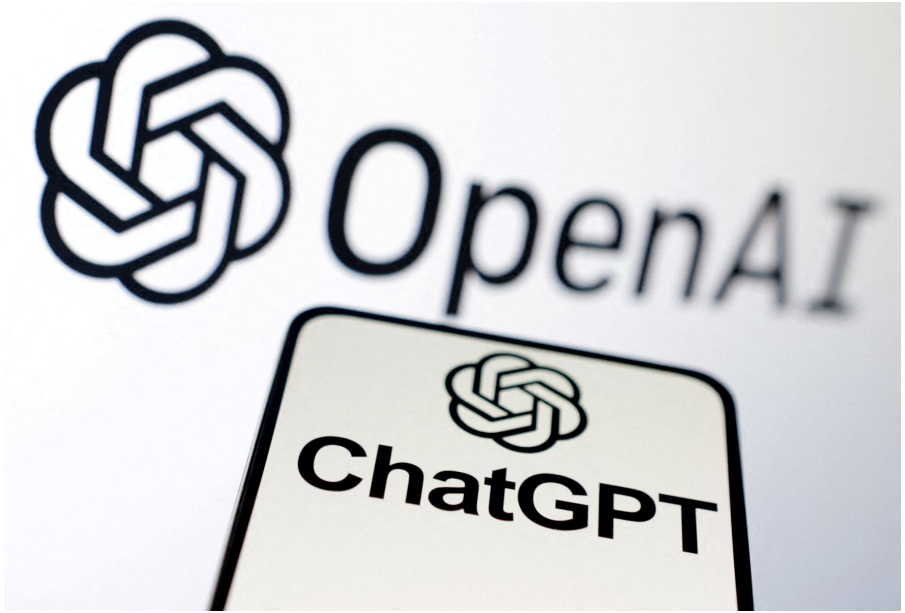
tgedatalabs.com

Explosion of Generative AI

Efficient Data Management: Vector databases are designed to efficiently store and manage large volumes of high-dimensional data.

Faster Training and Inference: Vector databases are optimized for searching and retrieving similar vectors quickly, significantly speeding up the training and inference process for generative AI models.

Enhanced Personalization and Relevance: Vector databases allow generative AI systems to offer more personalized and relevant content to users.



Efficient Similarity Search

Efficient Indexing: Vector Databases use advanced indexing techniques which allow for quick lookup and retrieval of similar vectors.

These indexes allow the database to quickly narrow down the search to a smaller, more relevant subset of vectors, significantly reducing search time.

Facilitating Complex Queries: Vector databases can handle complex queries that combine similarity with other search criteria.

This flexibility is increasingly important in advanced AI applications that require sophisticated search capabilities.

Query Image

Similar Images

Image Source

Popular Use Cases

- **Personalized recommendations:** E-commerce platforms use vector databases to recommend products to users based on their past behavior and preferences.
- **Fraud detection:** Financial institutions leverage vector search to identify anomalies and fraudulent transactions in real-time.
- **Scientific research:** Researchers use vector databases to analyze large datasets of molecules and genes, accelerating drug discovery and scientific progress.
- **Content moderation:** Social media platforms utilize vector databases to filter out harmful content like hate speech and misinformation.

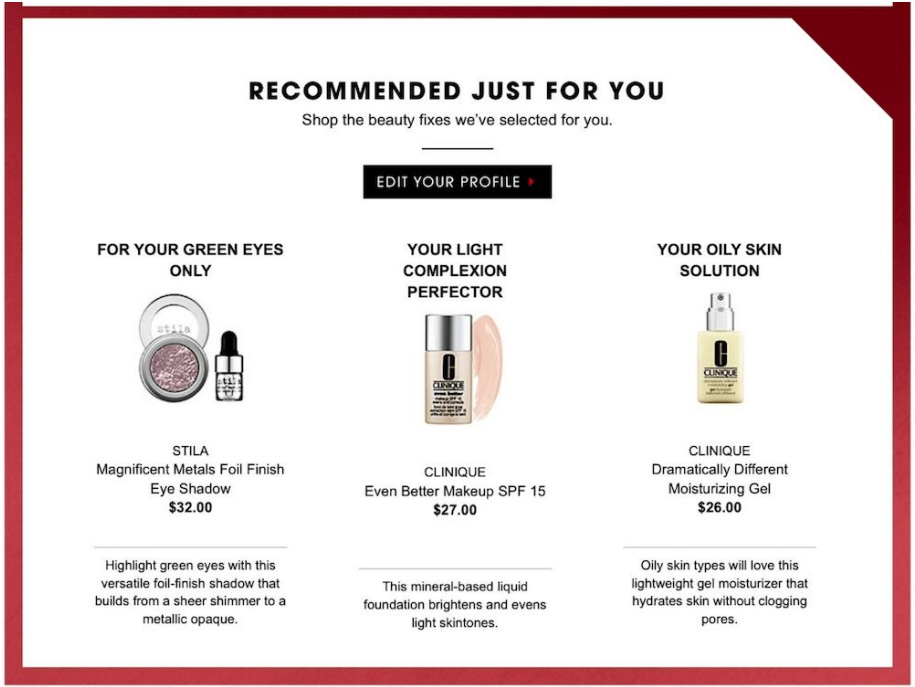


Image Source

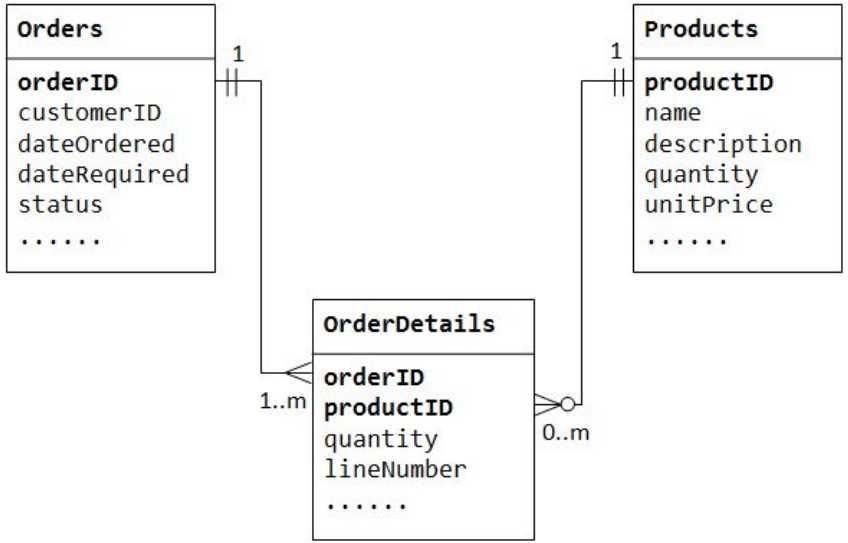
How Vector Databases differ from Traditional Databases

TGE
DATA LABS

tgedatalabs.com

Traditional Databases

- Traditional relational databases store data in structured tables, organizing data into rows and columns.
- Designed for general-purpose data storage and retrieval, with a focus on ACID (Atomicity, Consistency, Isolation, Durability) properties for transactional data integrity
- Suited for applications that require transaction processing, record-keeping, and reporting in businesses, finance, and other standard data management scenarios.



Vector Databases

- Vector databases are optimized for handling unstructured, high-dimensional data such as images, text documents, and user embeddings.
- They excel at similarity search and retrieval, enabling applications that require efficient search based on semantic similarity rather than exact keyword matches.
- Ideal for applications involving machine learning models, such as recommendation systems, image and voice recognition, and natural language processing.

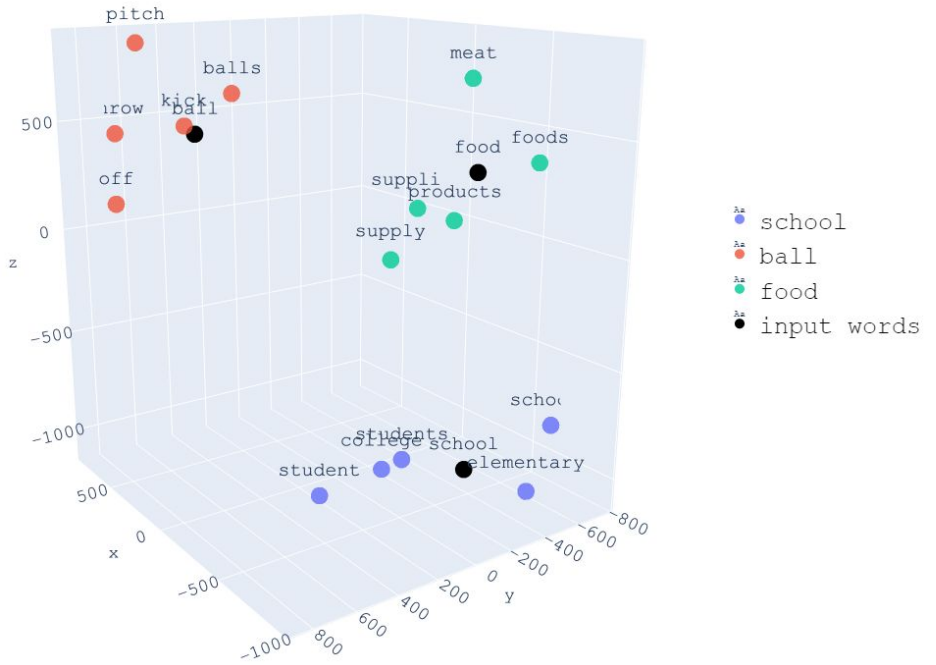


Image Source

Comparison

Feature	Traditional Database	Vector Database
Data Representation	Tables with rows and columns	Vectors
Querying	Exact Matches, range queries and joins	Similarity Search
Scalability & Performance	Designed for general-purpose data storage and retrieval, with a focus on ACID	Optimized for high-throughput and low-latency retrieval of complex vector data
Indexing & Search Efficiency	KD-trees, R-trees, approximate nearest neighbor (ANN) search algorithms	B-trees, hash indexes, full-text search indices

Advantages & Challenges

Advantages of Vector Databases

- **Efficient Similarity Search:** Vector databases excel at identifying similar data points based on their vector representations, enabling applications that rely on semantic similarity rather than exact keyword matches.
- **High-Dimensional Data Management:** Vector databases are designed to handle large volumes of high-dimensional data efficiently, making them suitable for modern data-intensive applications.
- **Scalability:** Vector databases can be scaled horizontally to accommodate increasing data volumes and processing demands.
- **Performance Optimization:** Vectorized algorithms accelerate search and retrieval operations, leading to faster response times.

Challenges of Vector Databases

- **Complexity in Understanding:** The concepts underlying vector databases can be complex, requiring a solid understanding of multi-dimensional data structures and algorithms.
- **Data Preprocessing:** Converting data into vector representations can be a complex and time-consuming process.
- **Resource Intensive:** High-dimensional vectors can be computationally expensive to process.
- **Query Optimization:** Optimizing vector database queries requires careful consideration of vector similarity metrics and search algorithms.

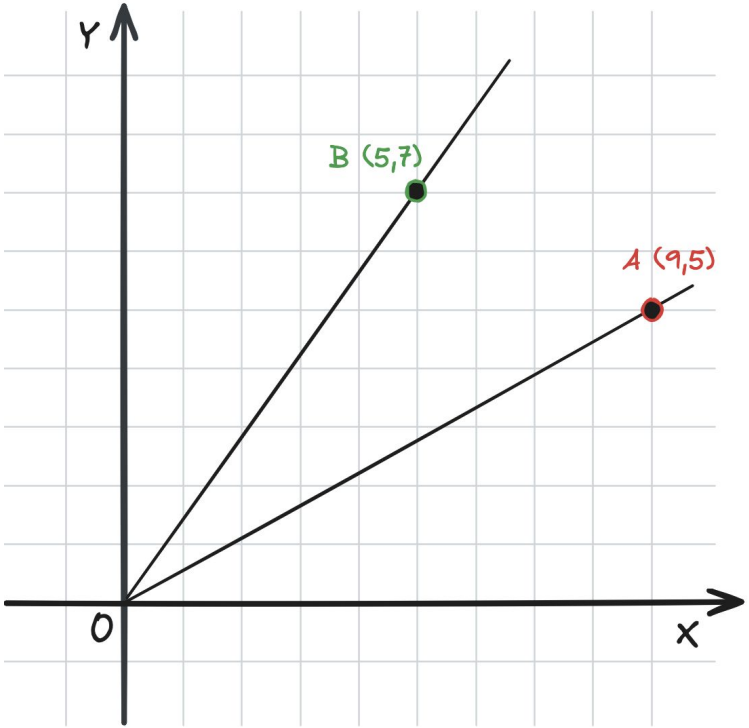
Vector Database Core Concepts

TGE
DATA LABS

tgedatalabs.com

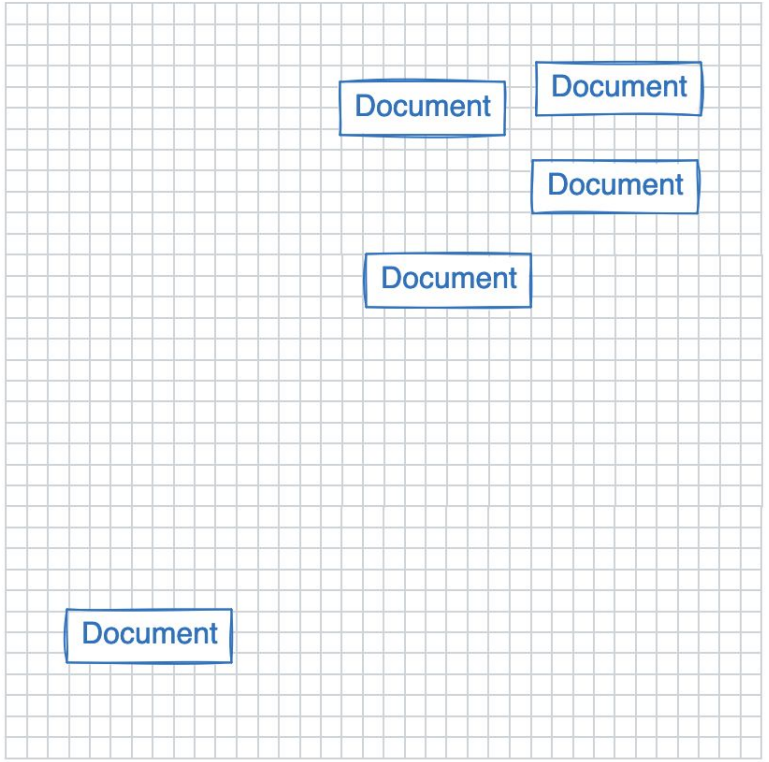
Introduction to Vectors

- Vectors are mathematical objects that represent quantities with both magnitude and direction.
- They can be represented as arrays of numbers, where each element corresponds to a specific dimension or component.
  - For example, representation of A is a (9, 5) while representation of B is (5, 7)
- In the context of databases, vectors are used to represent data points. Each data point is mapped to a vector, with its features or attributes represented by the components of the vector.



More on Vectors

- A vector represents data as a coordinate point in n-dimensional space. For example, a 2-dimensional vector plots a point on an x-y grid.
- Now imagine every entity in a database (users, products, documents, etc.) gets mapped to coordinates in a high-dimensional vector space.
- Vectors positioned closer together have greater similarity than those far apart.
- This spatial relationship allows precise queries about data similarity that would be extremely slow or impossible with legacy architectures.

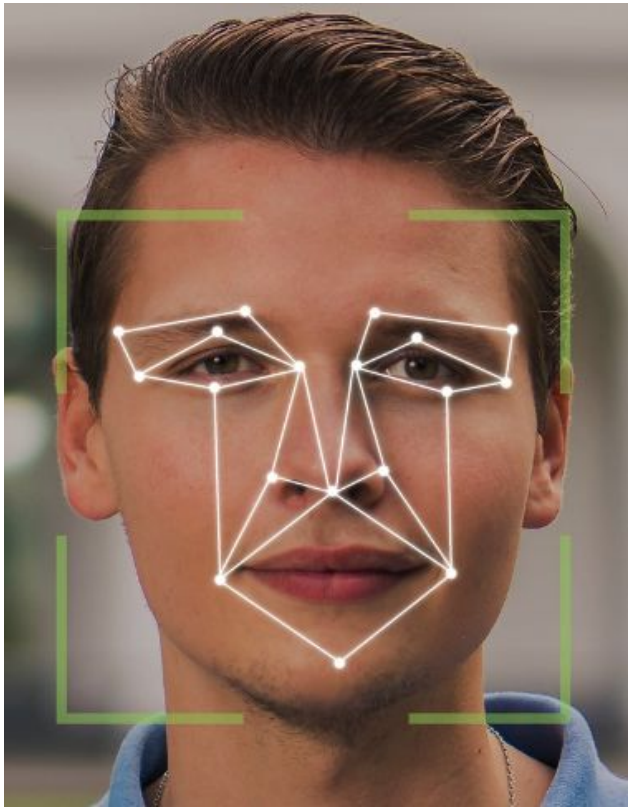


Real World Illustration - Supermarket Aisles

- Imagine you're in a supermarket looking for apples and any products made with apples (juice, candy apple, puree etc.)
- You can walk down each aisle (a dimension) and look at every product (a data point). This is the traditional database approach.
- But what if you could somehow "encode" the characteristics of apples (color, size, taste) into a single point in a multidimensional space
- That point would be a vector, representing the essence of "apple-ness" in a way that's independent of its location on the shelves

Real World Illustration - Facial Recognition

- Imagine each facial feature (eyes, nose, mouth) as a dimension in a high-dimensional space.
- The unique combination of these features for a specific person creates their facial vector.
- When you upload a picture, the system compares its vector to the stored vectors of known individuals.
- If it finds a close match, it recognizes the person in the picture!

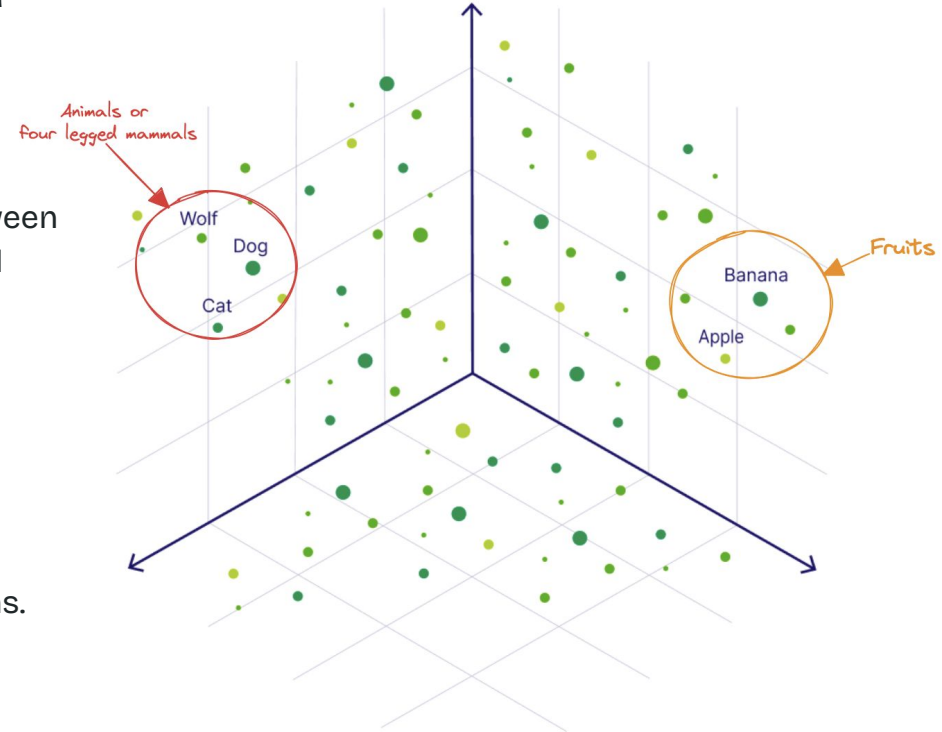


Vectors and their role in databases

- In vector databases, vectors represent complex data such as text, images, and sounds in a machine-understandable format.
- By representing data points as vectors, vector databases can quantify the semantic similarity between data points, allowing for applications that go beyond exact keyword matches.

Each element of a vector is a feature, and the entire vector encapsulates the essence of the data item.

This representation is ideal for similarity search because it transforms qualitative attributes into quantitative features that can be easily compared using mathematical operations.

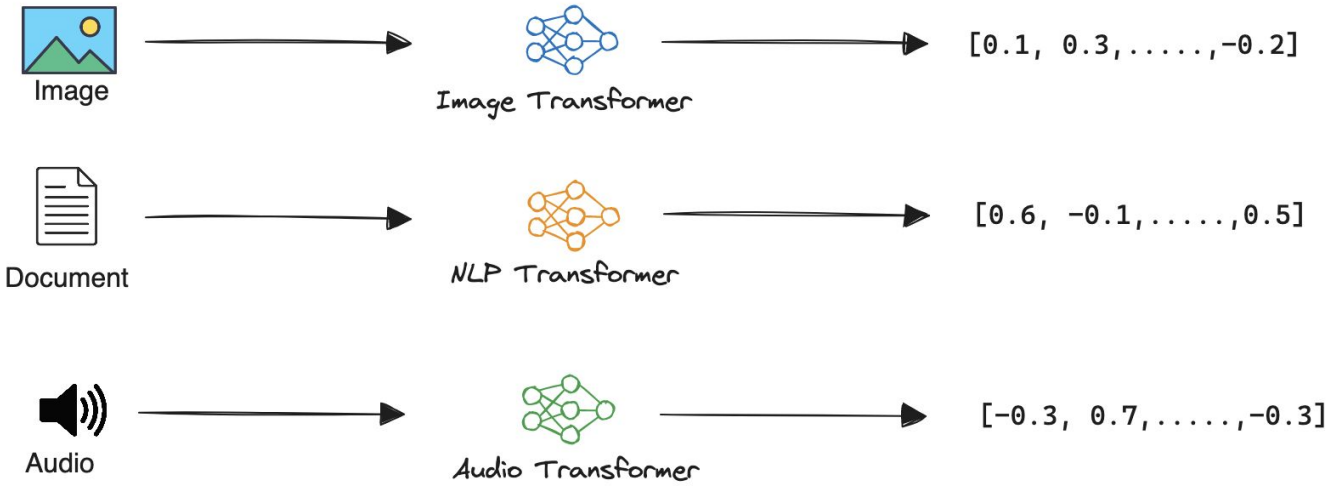


Introduction to Embeddings

Embeddings are vector representations of data points that capture their semantic meaning or relationships.

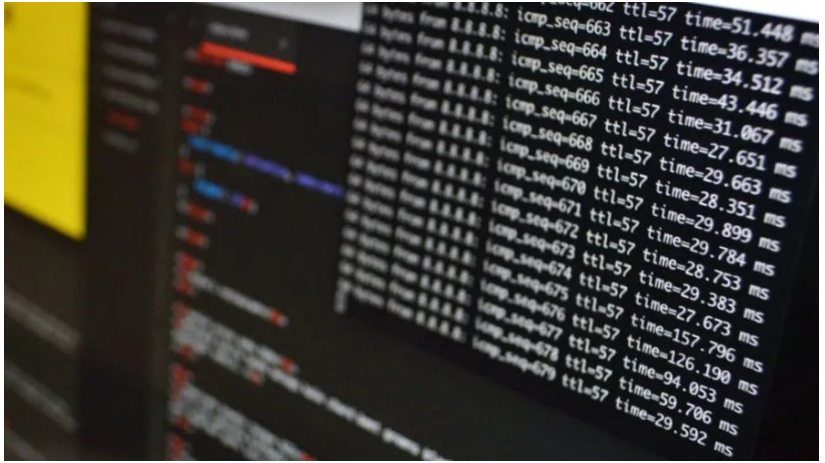
Data embeddings convert objects like users, products, documents, and more into dense numerical vectors that encode semantic meaning.

The resulting vectors represent capture semantic relationships, contextual usage and features



Embeddings Illustration - Fraud Detection

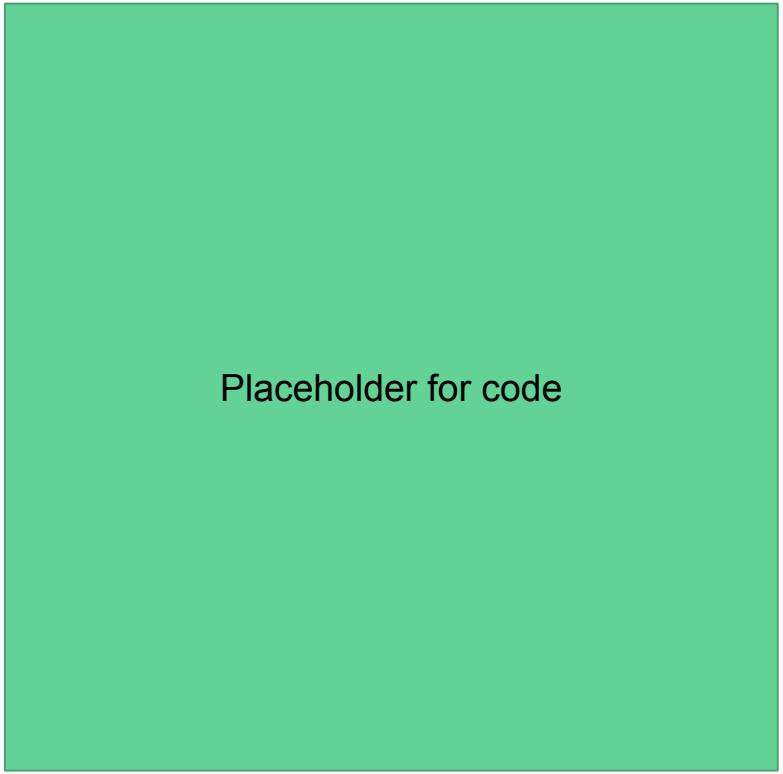
- Imagine a financial institution trying to identify fraudulent transactions. They analyze various data points like spending patterns, location, and device used
- Each data point is converted into a vector, forming a "transaction fingerprint" in the embedding space.
- Normal transactions cluster together, while fraudulent ones deviate significantly
- By monitoring for these outliers, the system can quickly detect and prevent suspicious activity.



Storing and Retrieving Embeddings

Embeddings are typically stored in vector spaces, allowing for efficient similarity search and retrieval operations.

- Vector Storage Optimization: Vector databases employ techniques like compression and indexing to optimize storage and retrieval of high-dimensional vectors.
- Similarity Search: Vector databases support efficient similarity search using various distance metrics and similarity measures.
- Embedding Management: Vector databases provide tools for managing and organizing embeddings, including metadata and versioning.



Dimensionality in Vector Spaces

TGE
DATA LABS

tgedatalabs.com

Introduction to Dimensionality and High-Dimensional Spaces

High-dimensional data refers to data that has a large number of features or attributes.

- A song can be described by its tempo, rhythm, genre, pitch, and even emotional tone.
- A picture isn't just pixels; it has color, brightness, texture, and even shapes and objects.

These datasets are usually hard to work with due to their complexity and volume of information that each data point contains.

As dimension counts increase, the volume of vector space grows exponentially.

High dimensionality allows for detailed and nuanced representation of complex data, but it also introduces difficulties in managing and searching this data, known as the "curse of dimensionality".

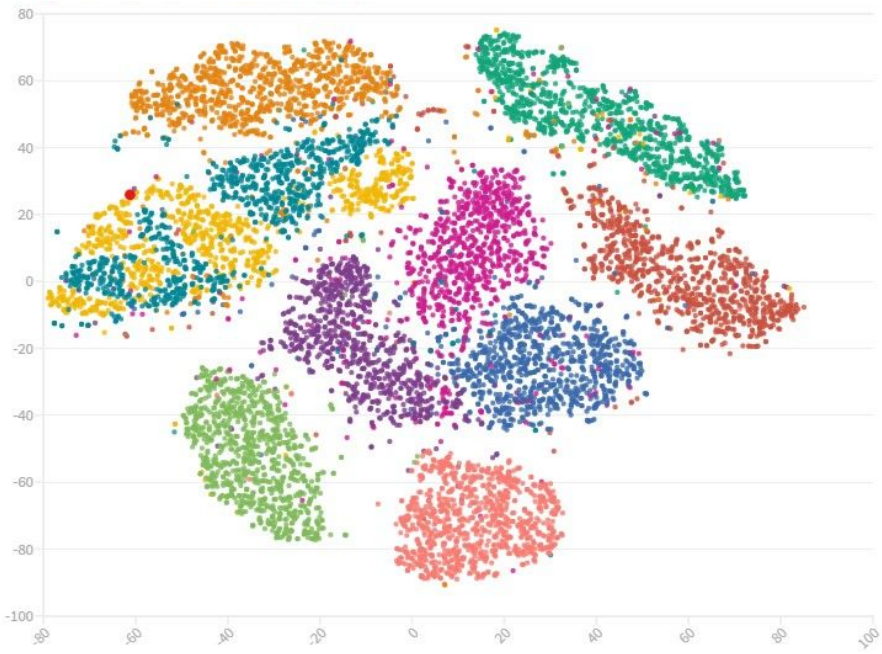


Image Source

Challenges with High-Dimensional Data

- **Curse of Dimensionality:** As the dimensionality increases, the distance between data points becomes less meaningful.
  The computational cost of searching through the data increases and the intuitive properties of distance that we rely on in lower dimensions no longer hold.

- **Computational Complexity:** Searching for similar points in high-dimensional spaces becomes computationally expensive due to the increasing number of possible distances to calculate.

Dimensionality Reduction

Dimensionality reduction techniques aim to reduce the number of dimensions in a vector representation while preserving the essential information. This can help alleviate the challenges associated with high-dimensional data.

- **Feature Selection:** Selecting a subset of relevant features that are most informative for the task. Dropping extraneous dimensions speeds computation significantly
- **Hashing:** Transforming high-dimensional vectors into lower-dimensional representations using hash functions.
- **Principal Component Analysis (PCA):** A statistical technique that identifies the principal components or directions of greatest variance in the data. This reduces the dataset's dimensionality while retaining most of the original variance

Distance Metrics and Similarity

Distance metrics are mathematical functions that determine the "distance" between two data points in a vector space. Different distance metrics can capture different aspects of similarity, making the choice of metric crucial for specific applications.

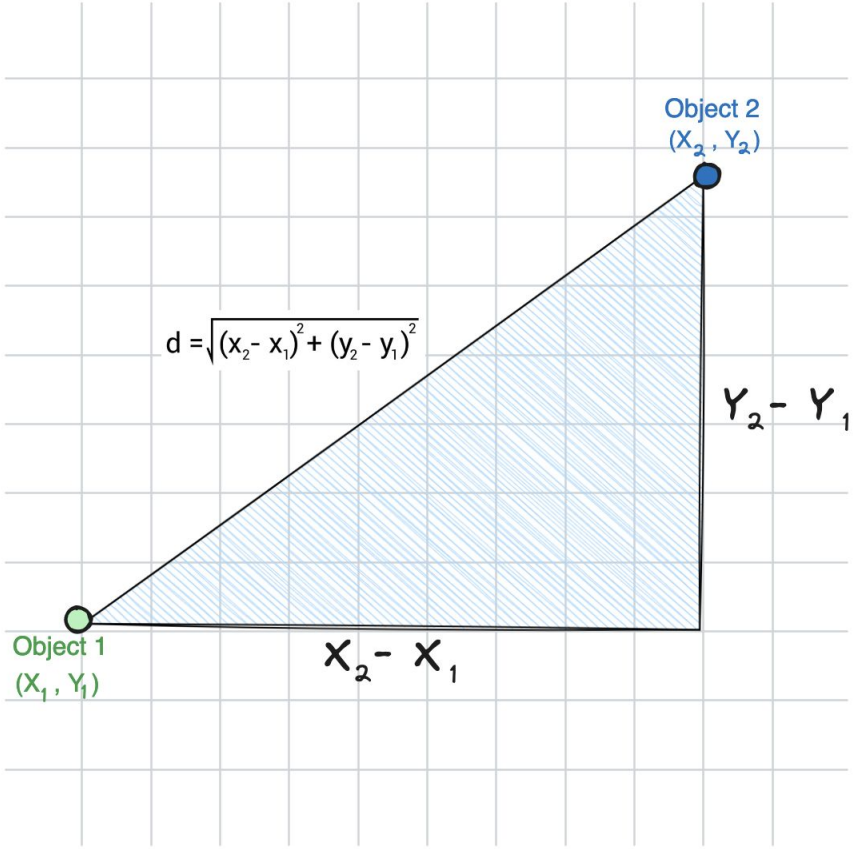
Distance metrics are mathematical tools used to define and measure the way we compute the similarity (or dissimilarity) between vectors. They are crucial for operations in vector databases

The most popular distance metrics are:

- ❏ Euclidean Distance
- ❏ Manhattan Distance
- ❏ Cosine Distance
- ❏ Jaccard Similarity

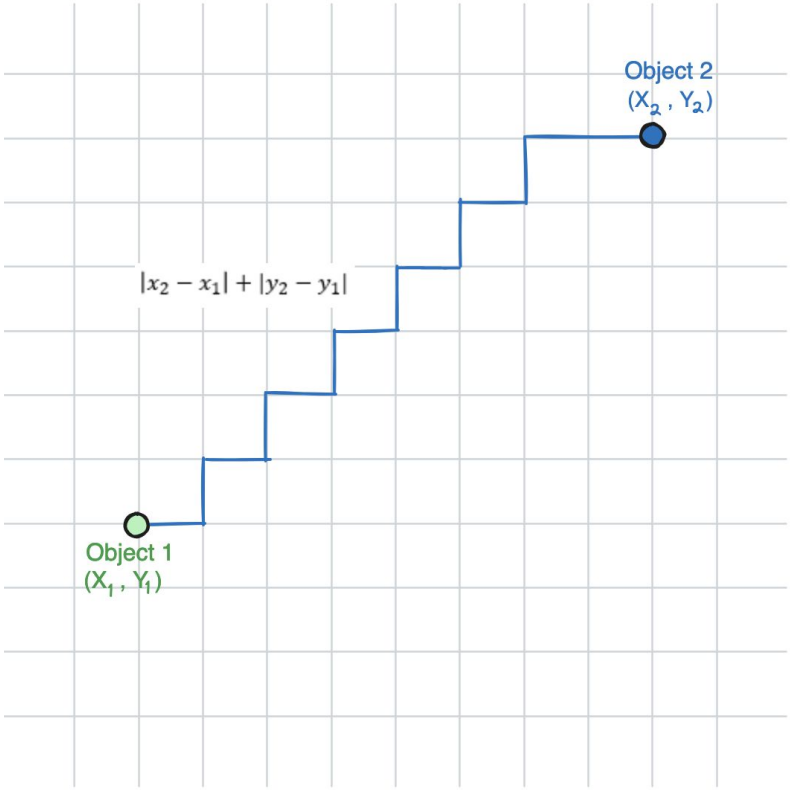
Euclidean Distance

- The most common distance metric
- Often referred to as L2 norm
- Measures the straight-line distance between two points in a vector space
- Very sensitive to the magnitude of vectors



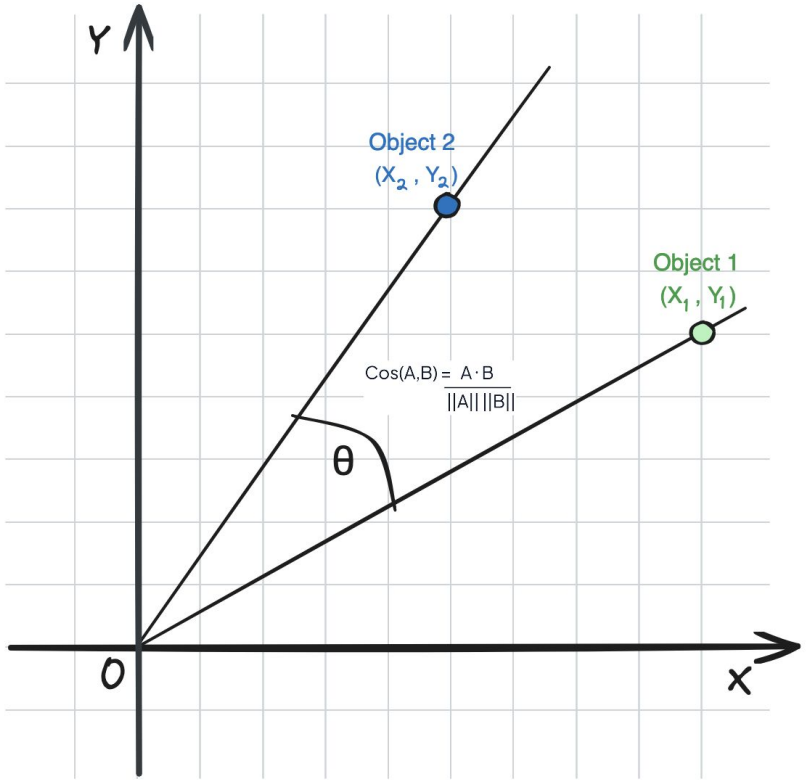
Manhattan Distance

- Also referred to as L1 norm or taxicab geometry
- Variant of Euclidean distance
- It sums the absolute differences of their corresponding coordinates
- Use Manhattan distance when you want to focus on how much each feature is different, not just how different they all are together.
- Often used for image retrieval and Financial Analysis



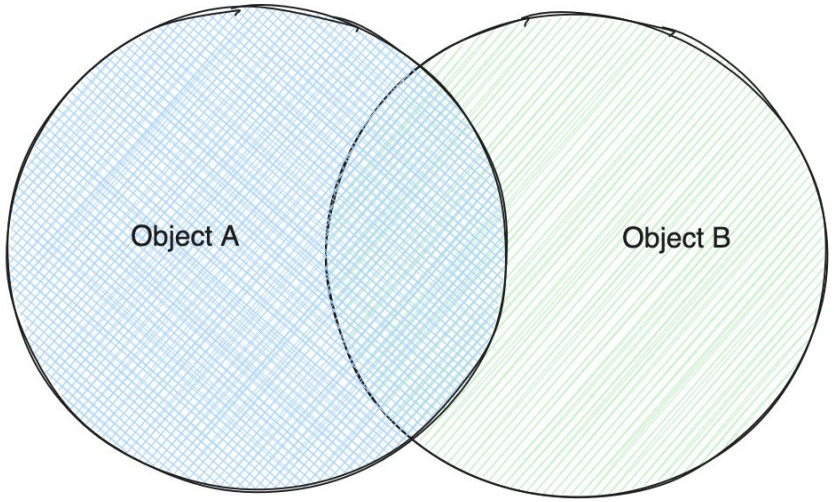
Cosine Distance

- Measures the cosine of the angle between two vectors where a higher cosine value indicates greater similarity
- Useful for text similarity where the frequency of words (term frequency) is less important than the angle or "direction" of the overall word
- For example, if a word appears 100 times in one document and 25 in another document, that is a difference in frequency of the word, but there is a chance that the documents are similar if we only consider the angle.



Jaccard Similarity

- Jaccard Similarity is used to compute the similarity between two objects, such as two text documents
- Measures overlap between sample sets to gauge similarity
- Example use cases:
  - Finding similar customers using their purchase history
  - Finding the similarity between two text documents using the number of terms used in both documents



$$\text{Jaccard} = \frac{\text{Intersection (A, B)}}{\text{Union (A, B)}}$$

Understanding Search Similarity

The core concept in Vector Databases

TGE
DATA LABS

tgedatalabs.com

The Importance of Search Similarity

Similarity search is the process of retrieving data points that are similar to a given query point based on a chosen distance metric or similarity measure.

In vector databases, it implies locating the closest vectors to the query vector as per a chosen distance metric.

When a user inputs a search query, the database converts it into a vector and uses distance metrics to find the data points (other vectors) that are closest to it.

This is fundamental to functionalities such as recommendation systems, where we wish to find items similar to a user's interests, or image search engines that retrieve images similar to a given image.

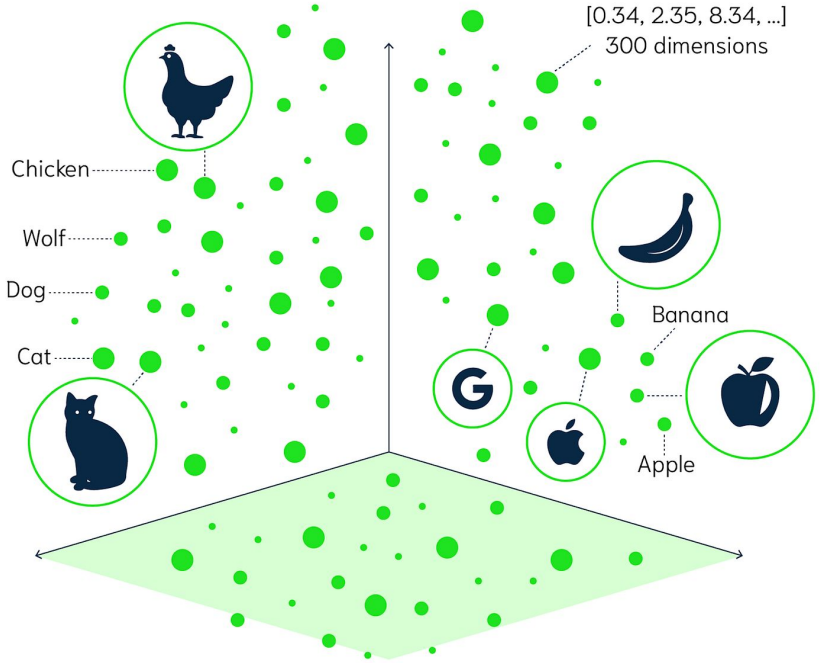


Image Source

K-Nearest Neighbors (KNN)

The k-nearest neighbors algorithm (KNN) is an algorithm used in both classification and regression tasks which operates on the principle that similar items exist close to each other

- "K" in KNN represents the number of nearest neighbors to consider. For example, if K=3, the algorithm looks at the three closest neighbors to a point.
- To find these neighbors, KNN calculates the distance between points using methods like Euclidean, Manhattan, or Hamming distance.
  - The choice of distance metric depends on the type of data.
- Advantages
  - Simple to understand and implement.
  - No need for complex feature engineering.
- Disadvantages
  - Can be computationally expensive for large datasets.
  - Sensitive to irrelevant features.
  - May suffer from the curse of dimensionality.

Approximate Nearest Neighbor (ANN)

Approximate Nearest Neighbors (ANN) is a technique used to efficiently find points in a large dataset that are similar, but not necessarily identical, to a given query point.

This is particularly useful in high-dimensional spaces, where traditional search methods become intractable due to the curse of dimensionality.

- Instead of searching the entire dataset for the exact nearest neighbors, ANN algorithms use various techniques to narrow down the search space and identify points that are likely to be close to the query request.

- This allows for significantly faster search times, especially in large datasets.

- Advantages
  - Faster search especially with large datasets
  - Easily scaled to handle massive datasets

- Disadvantages
  - ANN algorithms can be memory-intensive, which can limit their use on low-resource devices.

KNN vs. ANN

- KNN is great for lower dimensions BUT KNN is a Brute-Force Algorithm
- As the size of your dataset increases, KNN becomes much more computationally expensive as it is an exact search algorithm
- Needs to compute distance between the query and every vector in the database

- With ANN, you give up on some accuracy for speed
- Much more efficient for large datasets due to their ability to learn compact representations.
- Can achieve high accuracy by learning complex patterns in the data
- ANN is the preferred choice for real-time search due to its speed

Indexing and Querying

TGE
DATA LABS

tgedatalabs.com

Indexing Strategies

Just like traditional database indexes, vector indexes optimize data structure to rapidly narrow search spaces and speed up the retrieval of records.

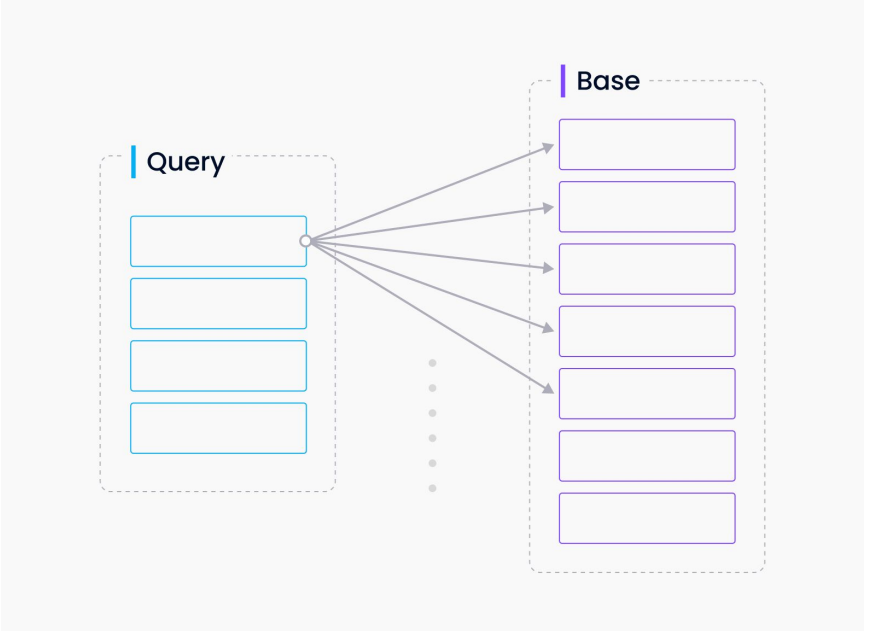
By creating indexes on specific dimensions or vector components, vector databases can significantly accelerate similarity search operations

Here are some popular indexing options:

- ❏ Flat Index
- ❏ Inverted File Index (IVF)
- ❏ Approximate Nearest Neighbors Oh Yeah (ANNOY)
- ❏ Product Quantization (PQ)
- ❏ Hierarchical Navigable Small World (HNSW)

Flat Index

- Flat indexes are ‘flat’ because it doesn't involve any advanced pre-processing or structuring of the data
- Instead, it stores the data points as they are, in a 'flat' structure, typically an array or a list.
- Flat Index prioritizes simplicity and direct access to data.
- Flat indexes work best for small document collections with limited vocabulary size
- They are fast to query but do not handle semantic matching or understand word meanings very well.
- Flat indices can be used as an initial step in a larger, more complex algorithmic process, particularly when data structuring is not yet defined.



Source

Flat Index - Imagined

- Imagine you have a large library with books organized by section - Fiction, Non-Fiction, Reference, etc.
- This is similar to a nested index, where you first go the section, then go to a specific shelf in that section to find your book.
- A flat index would be like if the library took all the books off the shelves and stacked them in one giant pile alphabetically by title.
- To find a book you just go to the spot in the giant stack where that book title falls alphabetically.
- This is much faster for locating books if you know the title, since you don't have to first navigate through sections and shelves.

Inverted File Index (IVF)

An inverted index is like a lookup table, used to find which data points (or documents) contain a particular value (or term).

It is a data structure used to map content to locations.

An inverted index consists of:

- A mapping of vector ID to location of vector. Allows retrieving vector given ID.
- An inverted list per dimension. Stores IDs of vectors having non zero entries in that dimension. Allows finding vectors that overlap in dimensions.

Doc	Text
1	Gold silver truck
2	Shipment of gold damaged in a fire
3	Delivery of silver arrived in a silver truck
4	Shipment of gold arrived in a truck

No.	Term	Times; Documents Words
1	a	<3; (2;6),(3;6),(4;6)>
2	arrived	<2; (3;4),(4;4)>
3	damaged	<1; (2;4)>
4	delivery	<1; (3;1)>
5	fire	<1; (2;7)>
6	gold	<3; (1;1),(2;3),(4;3)>
7	of	<3; (2;2),(3;2),(4;2)>
8	in	<3; (2;5),(3;5),(4;5)>
9	shipment	<2; (2;1),(4;1)>
10	silver	<2; (1;2),(3;3,7)>
11	truck	<3; (1;3),(3;8),(4;7)>

Term Dictionary | Posting List

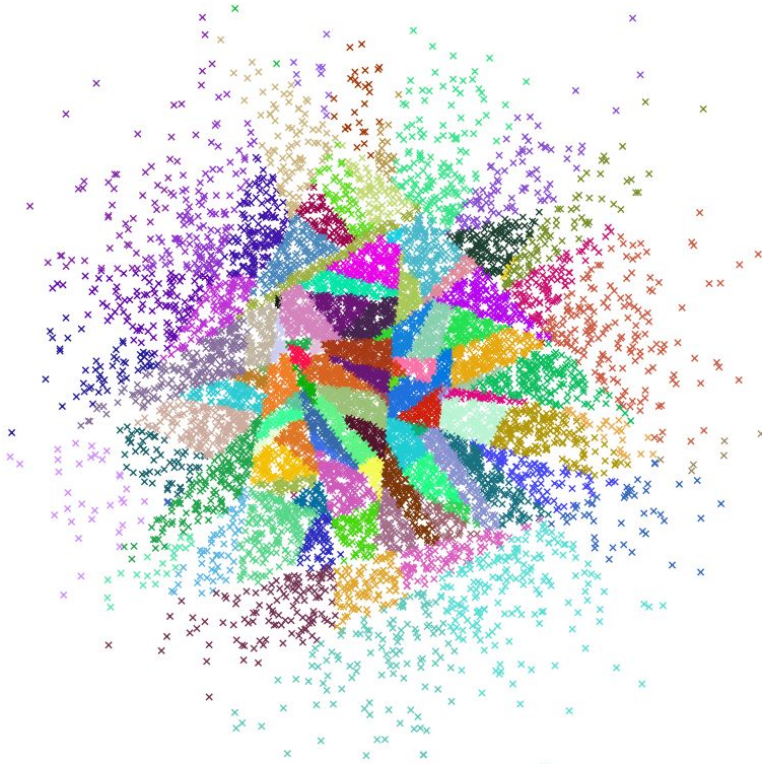
Source

Inverted File Index - Imagined

- Think of a newspaper office that publishes many articles every day.
- As each article is written, the editor adds metadata tags describing the key people, places, organizations and topics covered in the article
- Over time an inverted index is automatically built up that maps each metadata tag to the articles associated with it.
- For example, if an article profiles the new CEO of a company, it might have metadata tags for the person's name, the company name, and the tag "CEO". So in the inverted index, those 3 tags would all point to that article.
- A reporter wanting to quickly get up to speed on a company could search the inverted index for the company name tag and instantly get links to all published articles that mention and relate to that company.

Approximate Nearest Neighbors Oh Yeah (ANNOY)

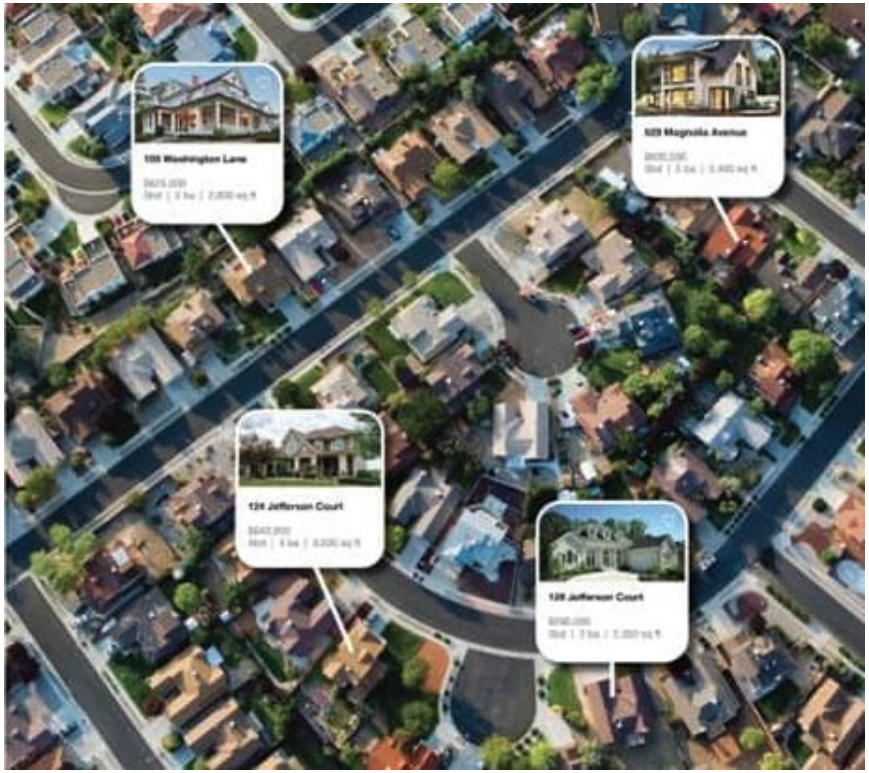
- Developed at Spotify, ANNOY works by constructing several trees (the forest) by recursively splitting the dataset along a randomly chosen axis.
- Each split is chosen to balance the tree as much as possible.
- Once the trees are built, ANNOY indexes them for efficient querying.
- In order to find the nearest neighbors of a query point, ANNOY searches through the trees. Instead of examining every single point, it traverses the trees, quickly narrowing down the candidate points.



Source

ANNOY - Imagined

- Imagine you have a large database of real estate listings with information on price, number of bedrooms, location, size etc.
- A potential buyer comes to you asking for recommendations similar to a house they liked
- ANNOY can create an index of all listings using selective search trees.
- When the buyer inquiry comes in, ANNOY does an approximate search through its prebuilt index to identify listings that may not be an exact match, but are similar in multiple attributes like price range, location and number of rooms.
- The search provides expanded options tailored to the buyer's interest even if there are no listings available that are precisely the same in every parameter.

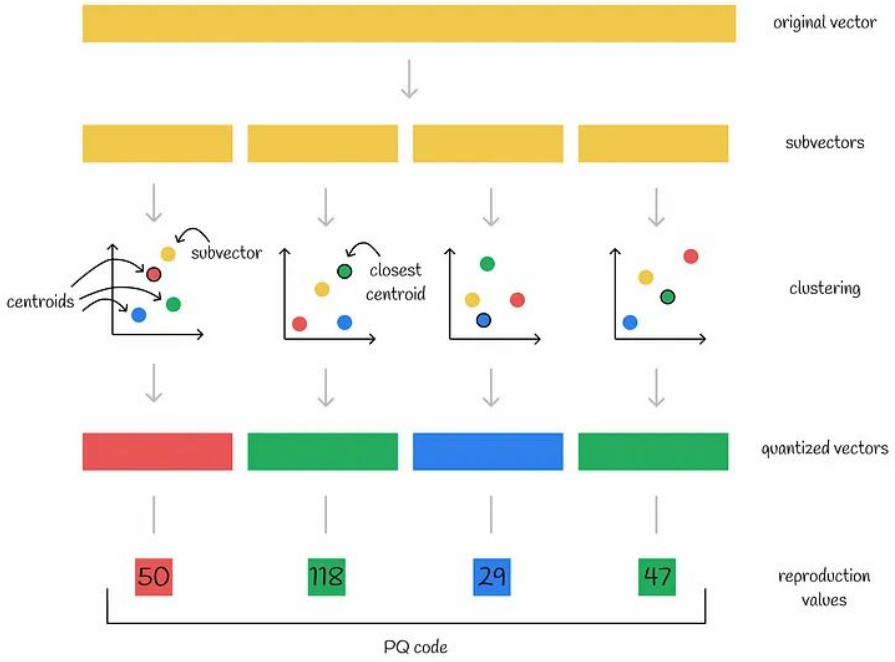


Product Quantization

Product Quantization is a method for partitioning a high-dimensional vector space by compressing them into shorter, more manageable representations.

It achieves this by:

- Splitting: Dividing the high-dimensional vector into smaller sub-vectors (segments).
- Quantization: Replacing each sub-vector with its nearest representative from a pre-defined codebook.
- This codebook essentially converts the continuous vector space into discrete counterparts, allowing efficient storage and comparison.
- Encoding: Combining the quantized representations of all sub-vectors into a single, compact code, known as the PQ code.



Source

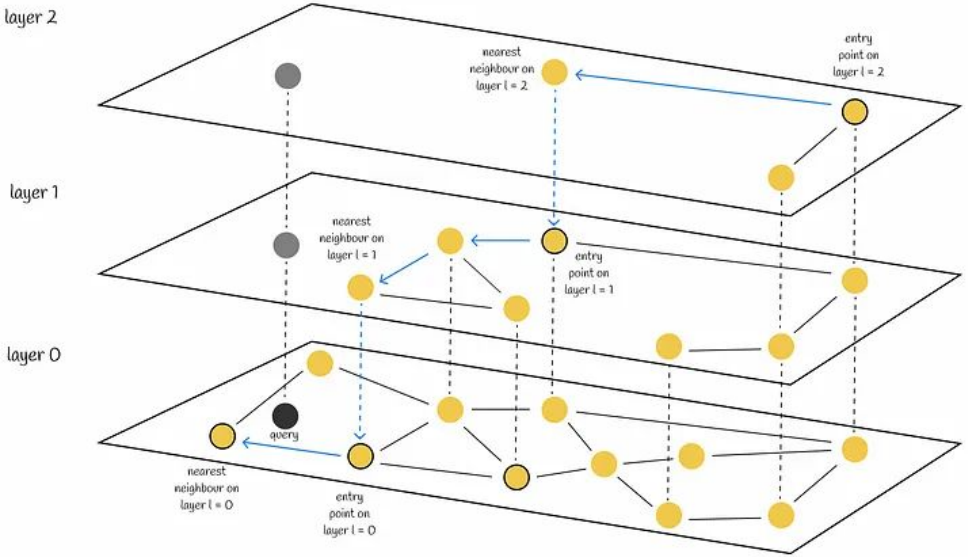
Product Quantization - Imagined

- Imagine you have a full closet overflowing with clothes, each with its own unique pattern, color, and style and you're trying to find the perfect fit
- Think of your clothes as data points in a high-dimensional space. Each dimension represents a different feature, like color, texture, and style. Product quantization helps you group these points efficiently by creating a simpler "codebook" for them.
- Pick a manageable number of "codewords."
  - "Fun outfits", "church outfits", "club outfits"
- Analyze each data point (clothes) and assign it to the closest codeword.
- Now, you can search for similar clothes by searching for their codewords.
- Instead of comparing every piece individually, you can quickly find items that share the same "code" based on features and style.

Hierarchical Navigable Small World (HNSW)

The HNSW algorithm is based on the concept of small world networks, where most nodes can be reached from every other by a small number of steps despite the network's large size.

- HNSW constructs a graph that connects similar vectors together while allowing efficient navigation between neighbors.
- HNSW enables fast and efficient nearest neighbor search, even for large datasets with millions or billions of data points.
- This graph structure can scale to massive databases with hundreds of millions or billions of vectors, while still supporting interactive speed searches.



HNSW - Imagined

- Imagine you're a tourist trying to find your way around a major tourist city
- Think of the city's landmarks and points of interest as data points. Each landmark has its unique features and location, like a museum's architectural style or a park's vibrant atmosphere
- Imagine HNSW as building map layers of interconnected "neighborhoods"
- These neighborhoods group nearby landmarks based on their similarities, like historical sites in one area and trendy cafes in another
- Within each neighborhood, shortcuts connect key landmarks directly. Imagine a hidden alleyway between two museums or a pedestrian bridge connecting parks.
- These shortcuts allow you to jump between nearby points without navigating every street.

Selecting the correct index

- Use **Flat** for small datasets where precision is critical.
- Use **IVF** for medium to large datasets where there's a balance between precision and speed.
- Use **HNSW** or **Annoy** for very large datasets where query speed is more important than precision.
- Use **PQ** when storage or memory is limited, as it provides a compressed representation.

Working with Vector Databases

(and Vector Stores)

TGE
DATA LABS

tgedatalabs.com

Before you decide: Vector Database or Vector Store?

When deciding to implement a Vector database within your organization, there is usually a debate between choosing a vector database or a vector store.

A vector database is a type of database designed from the ground up to efficiently store, index, and query vector data.

A vector store, in most cases, is a type of database used for general storage, search and retrieval but which also has the ability to store and retrieve vector data.

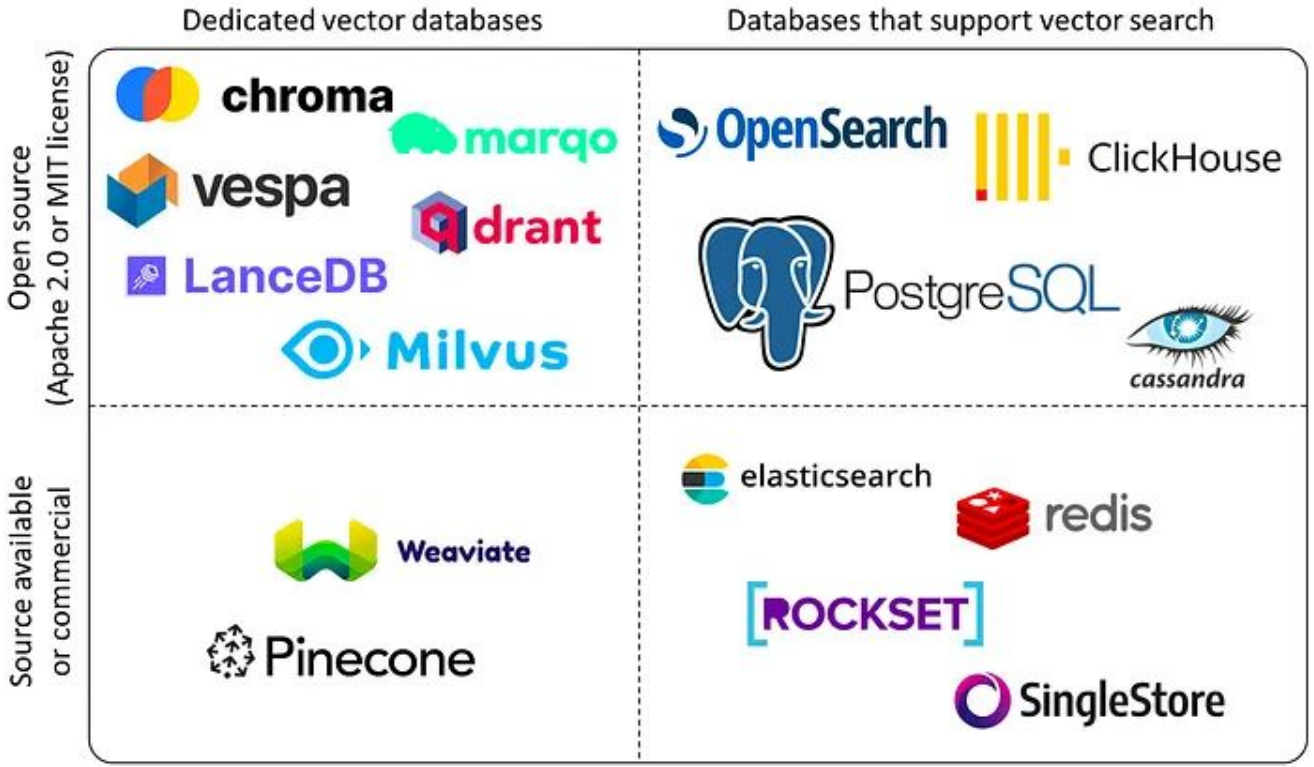
Vector Database Providers

- ❏ Pinecone
- ❏ Qdrant
- ❏ Milvus
- ❏ Weaviate
- ❏ Chroma

Vector Store Providers

- ❏ Elasticsearch
- ❏ Singlestore
- ❏ Typesense
- ❏ PostgreSQL

Do you need a Dedicated Vector Database?



Argument against a dedicated database

Pinecone

- Offers a cloud-native vector database optimized for speed and scalability

- Pinecone utilizes the concept of “indexes” and “pods”
  - Indexes are logical grouping of vectors (similar to a database table)
  - A pod is a containerized unit that exist within the index responsible for storing and managing vector data.
  - Pods range from a Standard (s1) pods for storage-intensive workloads to Performance (p1, p2) pods for demanding search queries.

- Pinecone uses a distributed architecture that can scale out horizontally across clusters while maintaining high availability.

www.pinecone.io

Qdrant

- Open-source vector search engine written in Rust
- Qdrant uses a concept called a "collection" to represent a set of related vectors. Each collection is divided into partitions for efficient scaling and distribution across shards.
- Qdrant offers both open-source and commercially licensed versions.
- Qdrant utilizes a raft consensus algorithm to ensure data consistency across shards.
- Dedicated libraries for Python and Go

www.qdrant.tech

Milvus

www.milvus.io

- Open-source vector database along with commercial hosted version (Zilliz)
- Milvus also uses "collections" to represent a collection of related vectors.
- Optimized for both analytics and search workloads
- Developers can customize metrics, index types, interfaces
- On-prem or cloud deployment

www.zilliz.com

Weaviate

- Vector search engine based on the semantic web
- Weaviate also uses "collections" (or “classes”) to represent a collection of related vectors.
- Weaviate uses GraphQL as its primary interface for interacting with the database
- Weaviate allows for the creation of custom modules that allows users to attach and run external models or code on data with Weaviate collections

Weaviate

www.weaviate.io

Demos

Pinecone Demo

https://colab.research.google.com/drive/1kzHNIRl4furh0ev_bEcf5pgwmCOhrMKd?usp=sharing

Weaviate Demo

https://colab.research.google.com/drive/11EKTjxyvnbh9ULusRV9_Tf0DAdxb36zz?usp=sharing

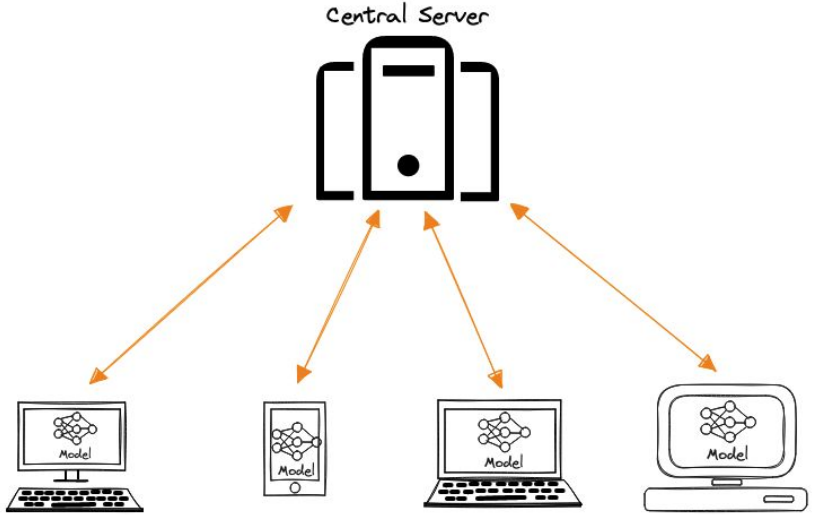
The Future of Vector Databases

TGE
DATA LABS

tgedatalabs.com

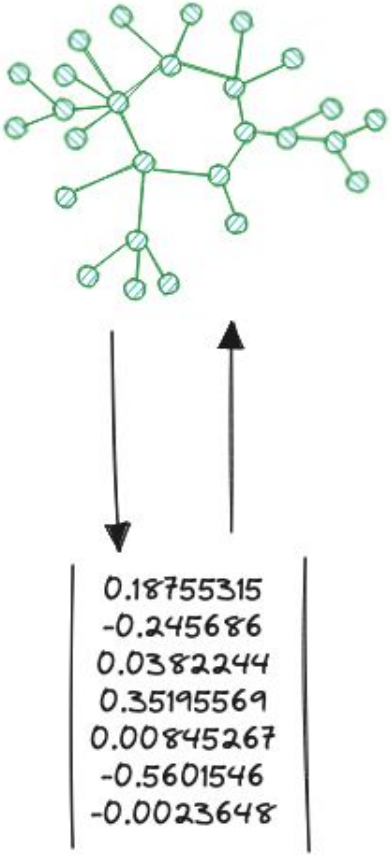
Federated Learning

- Federated Learning: Federated learning enables vector databases to collaboratively train and update embedding models without sharing sensitive data.
- This approach is particularly useful in privacy-sensitive applications, such as healthcare and finance.
- New possibilities for collaborative AI across borders and industries.



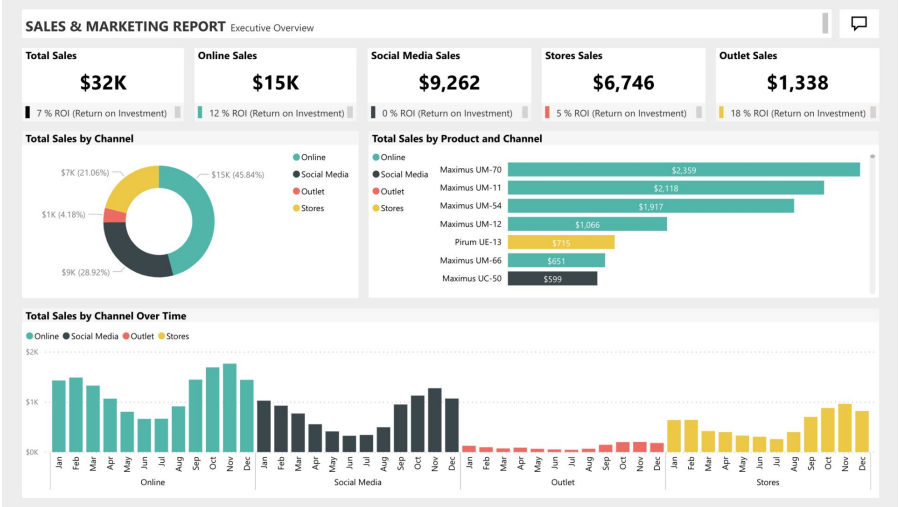
Graph Embeddings

- Graph Embeddings: Graph embeddings represent graphs as vectors, enabling similarity search and analysis of complex network data.
- This feature is expected to have a significant impact on fields like social network analysis and knowledge graph management.



BI Connectivity

- Integration of vector data into traditional reporting stacks opens easier adoption by leveraging existing SQL visualizations and dashboard tools while tapping backend speed.



Research Areas and Potential Future Applications

Research Areas I'm excited about:

- **Cross-modal Search:** Cross-modal search enables similarity search across different data modalities, such as text, images, and audio. This has the potential to revolutionize applications like multimedia retrieval and content recommendation.

- **Explainable AI:** Explainable AI techniques are being applied to vector databases to help users understand the reasoning behind similarity search results. This can enhance trust and transparency in AI-driven applications.

If you enjoyed the presentation, you'll love the video course

Access it here

For more information about our services, please reach out at
work@tgedatalabs.com

TGE DATA LABS

tgedatalabs.com