Best Vector Databases with Use Cases and Examples

Table of Contents

1. Introduction to Vector Databases

Vector databases are specialized database systems designed to store, manage, and index massive quantities of high-dimensional vector data efficiently. Unlike traditional databases that handle structured data in rows and columns, vector databases focus on high-dimensional data representations, often used in machine learning and AI applications[1] [2] [3].

Key Concepts

Vector Embeddings: These are dense representations of text, images, audio, or other data in a high-dimensional space where similar items are positioned close to each other. Each vector contains numerical values that capture the semantic meaning of the original data[4] [5].

Similarity Search: Vector databases enable finding data points that are closest to a given query vector using distance metrics like cosine similarity or Euclidean distance. This allows for semantic search that goes beyond keyword matching[4] [6].

High-Dimensional Indexing: Specialized indexing techniques like HNSW (Hierarchical Navigable Small World) or IVF (Inverted File) organize vectors for efficient similarity searches, even with millions or billions of vectors[4] [6].

Why Vector Databases Matter

The exponential growth of unstructured data and advances in AI/ML models generating embeddings have made vector databases essential for modern applications. They enable semantic understanding, pattern recognition, and similarity matching at scale, powering applications from recommendation systems to generative AI[7] [8].

2. How Vector Databases Work

Data Ingestion Process

1. **Data Preprocessing**: Raw data (text, images, audio) is cleaned and prepared for vectorization[9] [10].
2. **Embedding Generation**: Machine learning models like BERT, OpenAI's embeddings, or custom models convert data into vector representations[4] [9].
3. **Vector Storage**: Embeddings are stored in the database with associated metadata and unique identifiers[6].
4. **Index Creation**: Specialized indices are built to enable fast similarity searches[4] [6].

Query Process

1. **Query Vectorization**: User queries are converted to vectors using the same embedding model[9] [11].
2. **Similarity Search**: The database calculates distances between the query vector and stored vectors[4] [5].
3. **Result Ranking**: Results are ranked by similarity score and returned with associated metadata[6] [11].

Distance Metrics

- **Cosine Similarity**: Measures the cosine of the angle between vectors, ideal for text embeddings[4] [5]
- **Euclidean Distance**: Measures straight-line distance between points in vector space[4] [12]
- **Dot Product**: Efficient for normalized vectors and certain use cases[13]

3. Top Vector Databases in 2025

3.1 Pinecone

Overview: Pinecone is a fully managed, cloud-native vector database that offers seamless API integration and hassle-free infrastructure management[1] [3] [14].

Key Features:

- High-performance similarity search with sub-50ms latency[1] [3]
- Automatic scaling to handle billions of vectors[14] [3]
- Metadata filtering capabilities[1] [3]
- Real-time updates and indexing[3] [14]
- Enterprise-grade security and reliability[3]

Use Cases:

- Large-scale recommendation systems
- Semantic search applications
- AI-powered chatbots and virtual assistants
- Image and video similarity search[1] [3]

Companies Using: Microsoft, Accenture, Notion, HubSpot, Shopify, ClickUp, Gong, Zapier[3]

3.2 Milvus

Overview: Milvus is an open-source vector database designed for managing massive-scale unstructured data with support for multiple indexing methods and distributed architecture[1] [3] [15].

Key Features:

- Millisecond search on trillion-vector datasets[15]
- Multiple indexing algorithms (IVF, HNSW, Annoy)[1] [15]
- Horizontal scalability with distributed deployment[1] [15]
- Rich APIs and ecosystem integration[15]
- Built-in replication and failover mechanisms[15]

Use Cases:

- Computer vision applications
- Natural language processing
- Recommendation engines
- Content-based image retrieval[1] [3]

Companies Using: Salesforce, Zomato, Grab, IKEA, PayPal, Shell, Walmart, Airbnb[3]

3.3 Qdrant

Overview: Qdrant is a vector similarity search engine designed for high-dimensional data processing with extensive filtering support and real-time capabilities[1] [3] [15].

Key Features:

- Advanced filtering with JSON payload support[1] [15]
- Real-time vector updates[1] [3]
- Hybrid search combining dense and sparse vectors[15]
- Distributed deployment with sharding and replication[15]
- Vector quantization for reduced memory usage[15]

Use Cases:

- Semantic-based matching applications
- Neural network integration
- Real-time recommendation systems
- Advanced search with complex filtering[1] [3]

Companies Using: Discord, Johnson & Johnson, Perplexity, Mozilla, Bosch[3]

3.4 Chroma

Overview: Chroma is an AI-native open-source vector database designed to simplify the development of Large Language Model (LLM) applications[1] [3] [15].

Key Features:

- LangChain and LlamaIndex integration[1] [15]
- Storage of embeddings and metadata[15]
- Document and query embedding capabilities[15]
- Simple API for rapid development[1] [15]
- Modular and well-organized codebase[1]

Use Cases:

- LLM applications development
- Natural language processing projects
- Prototype and research environments
- Small to medium-scale deployments[16] [3]

3.5 Weaviate

Overview: Weaviate is a cloud-native, GraphQL-based vector database designed for large-scale, AI-powered applications with native ML integration[1] [3] [14].

Key Features:

- GraphQL interface for easy querying[3] [14]
- Native integration with ML models[1] [3]
- Schema flexibility and automatic vectorization[3] [14]
- Multi-modal search capabilities[3]
- Kubernetes-native deployment[1]

Use Cases:

- Multi-modal search applications
- Content management systems
- Knowledge graphs with vector search
- Enterprise AI applications[1] [3]

Companies Using: Red Hat, Stack Overflow, Mutiny, Red Bull, Writesonic[3]

3.6 PostgreSQL with pgvector

Overview: PostgreSQL extended with pgvector provides vector capabilities within a familiar relational database environment [3] [14] [17].

Key Features:

- Integration with existing PostgreSQL infrastructure [3] [14]
- Support for exact and approximate nearest neighbor search [14]
- ACID compliance and transactional guarantees [14]
- Cost-effective for existing PostgreSQL users [3]
- Strong consistency and durability [17]

Performance Highlights: Recent benchmarks show PostgreSQL with pgvector achieving 11.4x higher throughput than Qdrant at 99% recall on 50M embeddings [17].

4. Comprehensive Use Cases with Examples

4.1 Natural Language Processing (NLP)

Description: Vector databases store word embeddings and sentence vectors for semantic similarity, sentiment analysis, and contextual understanding [2] [18] [19].

Real-world Example:

- **Customer Support Chatbot**: When a user asks "How do I reset my password?", the system converts this to a vector and finds semantically similar queries like "Steps for password change" to provide relevant responses, even with different phrasing [2].
- **Document Similarity**: Legal firms use vector databases to find similar contracts or legal documents based on content rather than keywords [18] [19].

Technical Implementation:

```
# Example using sentence transformers
from sentence_transformers import SentenceTransformer
import qdrant_client

# Initialize model and client
model = SentenceTransformer('all-MiniLM-L6-v2')
client = qdrant_client.QdrantClient("localhost", port=6333)

# Vectorize and store documents
documents = ["How to reset password", "Password recovery steps", "Account login issues"]
embeddings = model.encode(documents)
```

4.2 Retrieval-Augmented Generation (RAG)

Description: RAG systems use vector databases to retrieve relevant context for Large Language Models, improving response accuracy and reducing hallucinations[10] [11] [9].

Process Flow:

1. **Ingestion**: Documents are chunked, embedded, and stored in vector database[9] [10]
2. **Retrieval**: User queries are vectorized and matched against stored embeddings[9] [11]
3. **Augmentation**: Retrieved context is combined with user query[9] [11]
4. **Generation**: LLM generates response using provided context[9] [11]

Real-world Example:

- **Enterprise Knowledge Base**: A company's internal documentation is embedded and stored. When employees ask questions, the system retrieves relevant documents and provides accurate, company-specific answers[10] [11].
- **Medical Assistant**: Healthcare providers use RAG with medical literature in vector databases to assist with diagnosis and treatment recommendations[9].

4.3 Image and Video Recognition

Description: Visual data is converted to high-dimensional vectors enabling similarity search, object detection, and content-based retrieval[2] [18] [19].

Real-world Example:

- **Pinterest**: When users pin images, the platform uses vector databases to suggest visually similar images, perhaps other beach landscapes or sunsets, enhancing content discovery[2].
- **Security Systems**: Airports use facial recognition systems where each face is converted to a vector and matched against databases of persons of interest[2].
- **E-commerce**: Fashion retailers enable "search by image" where users upload photos to find similar products[18].

4.4 Recommendation Systems

Description: User preferences and item attributes are vectorized to provide personalized recommendations based on similarity[18] [19] [20].

Real-world Example:

- **Netflix**: User viewing history and content attributes are vectorized to recommend movies and shows with similar characteristics[20].
- **Spotify**: Songs are represented as vectors using attributes like genre, tempo, and mood to recommend similar music[18].
- **Amazon**: Product features and user behavior create vectors for personalized product suggestions[20].

4.5 Anomaly and Fraud Detection

Description: Normal behavior patterns are stored as vectors, enabling quick identification of anomalous activities by measuring distance from typical patterns[2] [18] [19].

Real-world Example:

- **Banking**: Transaction patterns are vectorized. Unusual transactions (large international purchases for users who typically make small local purchases) are flagged as potential fraud[2].
- **Cybersecurity**: Network traffic patterns stored as vectors help identify potential intrusions or malicious activities[18] [19].
- **Manufacturing**: Sensor data from equipment is vectorized to detect anomalies indicating potential failures[19].

4.6 Semantic Search

Description: Search that understands meaning and context rather than just keyword matching, enabling more relevant results[19] [5] [21].

Real-world Example:

- **Legal Research**: Lawyers can search for "cases about intellectual property disputes" and find relevant cases even if they don't contain those exact terms[5] [19].
- **Scientific Literature**: Researchers can find papers on similar topics using conceptual queries rather than specific terminology[5].
- **E-commerce**: Customers can search for "comfortable running shoes for flat feet" and find products matching the intent[19].

4.7 Healthcare Applications

Description: Medical data vectorization enables patient similarity analysis, drug discovery, and clinical decision support[2] [18] [19].

Real-world Example:

- **Patient Similarity**: Hospital systems vectorize patient records (symptoms, medical history, genetics) to find similar cases and suggest effective treatments[2] [18].
- **Drug Discovery**: Pharmaceutical companies represent molecular structures as vectors to identify compounds with similar properties for drug development[2] [18].
- **Medical Imaging**: Radiology images are vectorized to assist in diagnosis by finding similar cases[18] [19].

4.8 Autonomous Vehicles

Description: Sensor data from cameras, LIDAR, and radar is processed into vectors for real-time navigation and obstacle detection[18] [19].

Real-world Example:

- **Self-driving Cars**: Vehicle systems convert environmental sensor data into vectors to identify objects, predict behavior, and make navigation decisions[18] [19].
- **Traffic Pattern Analysis**: City planners use vectorized traffic data to optimize signal timing and route planning[19].

4.9 Financial Services

Description: Portfolio analysis, risk assessment, and trading pattern recognition using vectorized financial data[2] [19].

Real-world Example:

- **Risk Assessment**: Banks vectorize loan applications and compare against historical data to assess default risk[19].
- **Trading Algorithms**: Investment firms use vector databases to identify similar market patterns and inform trading decisions[2] [19].
- **Portfolio Optimization**: Asset characteristics are vectorized to find optimal portfolio compositions[19].

4.10 Geospatial Applications

Description: Location data, routes, and geographical features vectorized for spatial analysis and optimization[2] [18].

Real-world Example:

- **Logistics Optimization**: Delivery companies like FedEx vectorize delivery points with location, traffic patterns, and priorities to optimize routes[2].
- **Urban Planning**: City governments vectorize urban regions with population density, building types, and land usage for development planning[2].
- **Navigation Systems**: GPS applications use vectorized map data for real-time route optimization[18].

5. Performance Benchmarks and Comparisons

5.1 Throughput Performance

Redis Performance: Recent benchmarks show Redis outperforming competitors with 62% more throughput than the second-ranked database for lower-dimensional datasets and 21% more for high-dimensional datasets[22].

PostgreSQL vs Specialized Databases: PostgreSQL with pgvector achieved 11.4x higher throughput than Qdrant at 99% recall when searching over 50M embeddings (471.57 QPS vs 41.47 QPS)[17].

5.2 Latency Comparisons

Query Latency Performance[22]:

- Redis: Up to 4x lower latency than Qdrant
- Redis: 4.67x lower latency than Milvus
- Redis: 1.71x lower latency than Weaviate

Qdrant vs PostgreSQL[17]:

- At 99% recall: Qdrant has 1% better p50 latency (30.75ms vs 31.07ms)
- At 90% recall: Qdrant has 50.3% lower p50 latency than PostgreSQL

5.3 Scalability Metrics

Vector Capacity[23]:

- Pinecone: 100B+ vectors supported
- Redis Vector: 10M vectors per node
- Milvus: Trillion-scale vector datasets[15]

Index Build Times[17]:

- Qdrant: 3.3 hours for 50M vectors
- PostgreSQL with pgvectorscale: 11.1 hours for 50M vectors

5.4 Comparative Analysis

Database	Type	Scalability	Performance	Ease of Use	Cost
Pinecone	Managed Service	Very High	Excellent	Very Easy	Paid ($70/month+) [23]
Milvus	Open Source DB	High	Good	Complex	Free
Qdrant	Open Source DB	High	Excellent	Moderate	Free
Redis	In-Memory DB	High	Excellent	Moderate	$40/month+ [23]
Weaviate	Open Source DB	High	Good	Moderate	Free
PostgreSQL+pgvector	SQL Extension	High	Very Good	Easy	Free

6. Cost Analysis and Optimization

6.1 Market Projections

The vector database market is experiencing rapid growth with an estimated 2025 market size of $500 million and a projected CAGR of 25% through 2033[7].

6.2 Cost Drivers

Primary Cost Factors[8]:

1. **Storage Costs**: Vector data storage requirements
2. **Compute Resources**: Query processing and indexing
3. **Query Volume**: Number of similarity searches performed
4. **Indexing Complexity**: Algorithm choice impacts resource usage
5. **Deployment Model**: Cloud vs on-premises costs

6.3 Cost Optimization Strategies

Performance Tuning[8]:

- **Optimize Indexing**: Choose appropriate algorithms (HNSW vs IVF)
- **Batch Queries**: Reduce overhead by grouping requests
- **Implement Caching**: Store frequently accessed results
- **Monitor Resource Usage**: Regular auditing of storage and compute

Open Source vs Commercial[8]:

- **Open Source Benefits**: Zero licensing costs, customization flexibility
- **Commercial Benefits**: Professional support, managed infrastructure, enterprise features

6.4 Real-world Cost Examples

E-commerce Recommendation System: Online retailer optimized indexing and query parameters, achieving 30% latency reduction and 20% cloud storage cost savings[8].

Healthcare Image Analysis: Hospital chose on-premises deployment with open-source tools, achieving 40% cost reduction compared to cloud solutions[8].

Financial Fraud Detection: Fintech company implemented batch queries and caching, reducing compute costs by 25% while maintaining accuracy[8].

7. Implementation Examples

7.1 Building a Semantic Search Engine

Using Qdrant with Python[6]:

```python
from qdrant_client import models, QdrantClient
from sentence_transformers import SentenceTransformer

# Initialize encoder and client
encoder = SentenceTransformer("all-MiniLM-L6-v2")
client = QdrantClient(":memory:")

# Create collection
client.create_collection(
    collection_name="my_books",
    vectors_config=models.VectorParams(
        size=encoder.get_sentence_embedding_dimension(),
        distance=models.Distance.COSINE,
    ),
)

# Upload data
documents = [
    {"name": "The Time Machine", "description": "A man travels through time..."},
    {"name": "Ender's Game", "description": "A young boy is trained for war..."}
]

client.upload_points(
    collection_name="my_books",
    points=[
        models.PointStruct(
            id=idx,
            vector=encoder.encode(doc["description"]).tolist(),
            payload=doc
        )
        for idx, doc in enumerate(documents)
    ],
)

# Query the database
hits = client.query_points(
    collection_name="my_books",
    query=encoder.encode("alien invasion").tolist(),
    limit=3,
).points
```

7.2 RAG Implementation with Vector Database

Spring AI Framework Example[24]:

```java
@Autowired
VectorStore vectorStore;

void load(String sourceFile) {
    JsonReader jsonReader = new JsonReader(
        new FileSystemResource(sourceFile),
        "price", "name", "shortDescription", "description", "tags"
    );
    List&lt;Document&gt; documents = jsonReader.get();
    this.vectorStore.add(documents);
}

// Query for similar documents
String question = &lt;user question&gt;
List&lt;Document&gt; similarDocuments = store.similaritySearch(this.question);
```

7.3 Multi-Modal Search Implementation

Using Weaviate for Image and Text Search[25]:

```
import weaviate

# Initialize client
client = weaviate.Client("http://localhost:8080")

# Create schema for multi-modal data
schema = {
    "classes": [{
        "class": "MultiModalContent",
        "properties": [
            {"name": "text", "dataType": ["text"]},
            {"name": "image", "dataType": ["blob"]},
            {"name": "category", "dataType": ["string"]}
        ],
        "vectorizer": "multi2vec-clip"
    }]
}

client.schema.create(schema)

# Query with mixed modalities
result = client.query.get("MultiModalContent", ["text", "category"]) \
    .with_near_text({"concepts": ["dog playing in park"]}) \
    .with_limit(5).do()
```

8. Choosing the Right Vector Database

8.1 Decision Framework

For Startups and Prototypes[16] [3]:

- **Chroma**: Simple setup, LangChain integration
- **FAISS**: Free, flexible, good for research
- **PostgreSQL+pgvector**: If already using PostgreSQL

For Production Applications[3] [16]:

- **Pinecone**: Managed service, enterprise features
- **Milvus**: Open source, highly scalable
- **Qdrant**: Advanced filtering, real-time updates

For Enterprise Deployments[3]:

- **Pinecone**: Full management, enterprise support
- **Weaviate**: GraphQL interface, ML integration
- **Milvus**: Open source with enterprise features

8.2 Key Evaluation Criteria

Performance Requirements[26]:

- Query latency needs (< 10ms, < 100ms)
- Throughput requirements (QPS)
- Dataset size and growth projections
- Recall/precision requirements

Operational Considerations[26]:

- Development team expertise
- Infrastructure management preferences
- Integration requirements
- Budget constraints
- Support needs

8.3 Hybrid Approaches

General-Purpose + Vector Extension[27] [17]:

- MongoDB Atlas with vector search
- PostgreSQL with pgvector
- Redis with vector capabilities
- SingleStore with vector support

Benefits: Unified data model, existing infrastructure leverage, ACID compliance

9. Future Trends and Recommendations

9.1 Emerging Trends

Technology Advancements[7] [19]:

- GPU acceleration becoming standard
- Hybrid search combining semantic and keyword search
- Improved indexing techniques (hierarchical clustering)
- Real-time feedback loops for adaptive learning

Market Evolution[7]:

- Convergence with other database types
- Cloud-native solutions dominating
- Specialized hardware acceleration
- Federated and distributed vector search

9.2 Industry Predictions

Market Growth[7]:

- 25% CAGR through 2033
- Asia-Pacific showing rapid adoption
- NLP applications driving primary growth
- Commercial databases maintaining market leadership

Technical Developments:

- More efficient quantization methods
- Advanced filtering capabilities
- Multi-modal search standardization
- Integration with generative AI pipelines

9.3 Strategic Recommendations

For Organizations Starting with Vector Databases:

1. Begin with proof-of-concept using open-source solutions
2. Evaluate performance requirements early
3. Consider future scalability needs
4. Invest in team training and expertise development

For Scaling Implementations:

1. Implement monitoring and observability
2. Plan for data growth and query volume increases
3. Consider hybrid architectures for different workloads

4. Establish clear cost optimization strategies

For Enterprise Deployments:

1. Prioritize managed services for reduced operational overhead
2. Implement comprehensive security and compliance measures
3. Plan for multi-region deployments
4. Establish vendor relationships and support contracts

Conclusion

Vector databases represent a fundamental shift in how we handle and query high-dimensional data, enabling sophisticated AI applications that were previously impossible or impractical. The landscape offers diverse options from fully managed services like Pinecone to open-source solutions like Milvus and Qdrant, each with distinct advantages for different use cases.

Key takeaways for implementation success:

1. **Match the solution to your needs**: Consider scalability, performance, and operational requirements
2. **Start simple and iterate**: Begin with prototypes using accessible tools like Chroma or FAISS
3. **Plan for growth**: Choose solutions that can scale with your data and query volume
4. **Invest in expertise**: Vector database success requires understanding of embeddings and similarity search
5. **Monitor and optimize**: Implement comprehensive observability for performance and cost optimization

The future of vector databases is bright, with continued innovation in performance, usability, and integration capabilities. Organizations that invest early in building vector search capabilities will be well-positioned to leverage the growing ecosystem of AI applications and services.

As the market continues to mature, we expect to see further convergence between traditional and vector databases, making these powerful capabilities accessible to a broader range of applications and use cases. The key is to start experimenting now and build the foundational knowledge that will enable success in this rapidly evolving space.
[28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51]

⁂

1. https://www.geeksforgeeks.org/dbms/top-vector-databases/
2. https://research.aimultiple.com/vector-database-use-cases/
3. https://www.cloudraft.io/blog/top-5-vector-databases
4. https://www.geeksforgeeks.org/data-science/implementing-semantic-search-with-vector-database/
5. https://www.kdnuggets.com/semantic-search-with-vector-databases
6. https://qdrant.tech/documentation/beginner-tutorials/search-beginners/
7. https://www.datainsightsmarket.com/reports/vector-database-1990919
8. https://www.meegle.com/en_us/topics/vector-databases/vector-database-cost-analysis
9. https://www.pinecone.io/learn/retrieval-augmented-generation/

10. https://objectbox.io/retrieval-augmented-generation-rag-with-vector-databases-expanding-ai-capabilities/
11. https://aws.amazon.com/what-is/retrieval-augmented-generation/
12. https://www.decube.io/post/vector-database-concept
13. https://learn.microsoft.com/en-us/azure/cosmos-db/vector-database
14. https://lakefs.io/blog/12-vector-databases-2023/
15. https://www.instaclustr.com/education/vector-database/top-10-open-source-vector-databases/
16. https://www.shakudo.io/blog/top-9-vector-databases
17. https://www.tigerdata.com/blog/pgvector-vs-qdrant
18. https://lakefs.io/blog/what-is-vector-databases/
19. https://www.instaclustr.com/education/vector-database/vector-database-13-use-cases-from-traditional-to-next-gen/
20. https://nexla.com/ai-infrastructure/vector-databases/
21. https://www.instaclustr.com/education/vector-database/vector-search-vs-semantic-search-4-key-differences-and-how-to-choose/
22. https://redis.io/blog/benchmarking-results-for-vector-databases/
23. https://aloa.co/ai/comparisons/vector-database-comparison/pinecone-vs-redis-vector
24. https://docs.spring.io/spring-ai/reference/api/vectordbs.html
25. https://blog.logrocket.com/implement-vector-database-ai/
26. https://www.linkedin.com/pulse/vectordbs-comparison-pros-cons-jimmy-wang-scjqc
27. https://www.reddit.com/r/LangChain/comments/170jigz/my_strategy_for_picking_a_vector_database_a/
28. https://www.pinecone.io/learn/vector-database/
29. https://www.reddit.com/r/MachineLearning/comments/1ijxrqj/whats_the_best_vector_db_whats_new_in_vector_db/
30. https://docs.weaviate.io/weaviate/more-resources/example-use-cases
31. https://www.yugabyte.com/key-concepts/top-five-vector-database-and-library-options-2025/
32. https://aws.amazon.com/what-is/vector-databases/
33. https://zilliz.com/blog/top-5-open-source-vector-search-engines
34. https://qdrant.tech/use-cases/
35. https://docs.mistral.ai/guides/rag/
36. https://learn.microsoft.com/en-us/shows/generative-ai-for-beginners/retrieval-augmented-generation-rag-and-vector-databases-generative-ai-for-beginners
37. https://www.tigerdata.com/learn/vector-search-vs-semantic-search
38. https://python.langchain.com/docs/tutorials/rag/
39. https://www.youtube.com/watch?v=Gxs3vPKxr-U
40. https://writer.com/engineering/rag-vector-database/
41. https://github.com/topics/vector-database
42. https://nextbrick.com/exploring-open-source-vector-search-engines-faiss-vs-milvus-vs-pinecone-2/
43. https://nextbrick.com/exploring-open-source-vector-search-engines-faiss-vs-milvus-vs-pinecone/
44. https://benchant.com/blog/single-store-vector-vs-pinecone-zilliz-2025

45. https://qdrant.tech/benchmarks/
46. https://www.reddit.com/r/vectordatabase/comments/170j6zd/my_strategy_for_picking_a_vector_databas
e_a/
47. https://benchmark.vectorview.ai/vectordbs.html
48. https://liquidmetal.ai/casesAndBlogs/vector-comparison/
49. https://superlinked.com/vector-db-comparison
50. https://www.turing.com/resources/vector-database-comparison
51. https://www.youtube.com/watch?v=o1spvLEiqUg