Day94_ChromaDB_VectorDB_and_CustomEmbeddings

October 1, 2025

Vector Databases ChromaDB

1 What is a Vector Database?

A **Vector Database** is a special type of database designed to store and search **vector embeddings** (numerical representations of text, images, or other data).

Unlike traditional databases that search using exact matches or SQL queries, vector databases allow **semantic search** — searching by meaning, not exact words.

Example:

- Query: *"latest iPhone release date"*
- Traditional DB → searches for rows with the exact words "iPhone release date".
- Vector DB → finds documents *about Apple products, launches, new iPhones*, even if the words don't match exactly.

2 How Vector Databases Work?

1. **Embedding Generation**
   - Raw data (text, image, audio) is converted into a vector (list of numbers) using ML models like `sentence-transformers`, `OpenAI embeddings`, `HuggingFace models`.

   Example:

   `"Apple is a fruit"` → `[0.12, -0.43, 0.98, ...]`
2. **Storage in Vector DB**
   - These vectors are stored in a database optimized for similarity search.
3. **Querying**
   - Query text/image is also converted into an embedding.
   - The DB uses **nearest neighbor search (ANN: Approximate Nearest Neighbor)** to find vectors closest in meaning.
4. **Return Results**

- The DB retrieves the most relevant items (documents, images, etc.).

3 Popular Free Vector Databases

- **ChromaDB** → Lightweight, Python-first, easy to integrate.
- **FAISS (Facebook AI Similarity Search)** → Highly optimized, local use.
- **Weaviate** → Open-source, has cloud support.
- **Pinecone (Freemium)** → Managed cloud service.
- **Milvus** → High-scale open-source solution.

  All have similar functionality (store vectors, perform similarity search). The main difference is **syntax and scalability**.
  Always check the **official docs** when stuck:

Documentation Links

- Sentence Transformers Documentation
- ChromaDB Embedding Functions Guide
- ChromaDB GitHub Repository

4 Installing Dependencies

```
# Install ChromaDB (specific version)
!pip install chromadb==0.5.3

# Upgrade HuggingFace Hub (for embeddings)
!pip install --upgrade huggingface_hub

# Upgrade Sentence Transformers (for embeddings)
!pip install --upgrade sentence-transformers
```

5 Using ChromaDB – Step by Step

[1]:
```
import chromadb

#  Create a Chroma client (in-memory database for now)
client = chromadb.Client()

# Create a collection (like a table in SQL)
# Note: A collection stores embeddings + documents.
collection = client.create_collection(name="news")

# Trying to create the same collection again will raise an error:
# collection = client.create_collection(name="news")
#  Error: Collection already exists

# To delete and recreate:
client.delete_collection("news")
collection = client.create_collection(name="news")

#  Insert documents into the collection
collection.add(
    documents=[
        "This is a document about Pineapple",
        "This is a document abount Apple"
    ],
    ids=["id1", "id2"]  # unique identifiers
)

#  Retrieve collection object (alternative way)
collection = client.get_collection("news")

#  Query: Search by meaning
results = collection.query(
    query_texts=["New iPhone will launch in May"],
    n_results=2
)

print("Search Results for iPhone query:")
print(results)
```

```
Search Results for iPhone query:
{'ids': [['id2', 'id1']], 'embeddings': None, 'documents': [['This is a document
abount Apple', 'This is a document about Pineapple']], 'uris': None, 'included':
['metadatas', 'documents', 'distances'], 'data': None, 'metadatas': [[None,
None]], 'distances': [[1.5486209392547607, 1.8884873390197754]]}
```

6 Output Example (Semantic Search)

Even though the query is about **iPhone**, the DB will return documents about **Apple** (the company/fruit) because of semantic similarity.

[3]:
```
{
 'ids': [['id2', 'id1']],
 'documents': [['This is a document abount Apple',
                'This is a document about Pineapple']],
 'distances': [[1.54, 1.88]]  # smaller = more similar
}
```

[3]:
```
{'ids': [['id2', 'id1']],
 'documents': [['This is a document abount Apple',
   'This is a document about Pineapple']],
 'distances': [[1.54, 1.88]]}
```

7 Another Query Example

Even though "Kiwi" is not in documents, it may find **Pineapple** or **Apple** because they are fruits → semantic match.

[4]:
```
results = collection.query(
    query_texts=["I just love KIWI"],
    n_results=2
)

print("Search Results for KIWI query:")
print(results)
```

```
Search Results for KIWI query:
{'ids': [['id2', 'id7']], 'embeddings': None, 'documents': [['Pineapple farming
has increased in tropical countries due to rising demand.', 'Tourists are
flocking to Japan to see cherry blossoms this spring.']], 'uris': None,
'included': ['metadatas', 'documents', 'distances'], 'data': None, 'metadatas':
[[None, None]], 'distances': [[1.706491470336914, 1.7645626068115234]]}
```

8 See Embeddings

This shows vector representations (long list of numbers) generated for each document.

[5]:
```
# To see stored embeddings
collection.peek()
```

[5]:
```
{'ids': ['id1',
  'id2',
  'id3',
  'id4',
  'id5',
  'id6',
  'id7',
  'id8',
  'id9',
  'id10'],
 'embeddings': array([[ 0.01253181,  0.00910238,  0.07303503, …, -0.05493123,
          0.06227031, -0.00570446],
        [ 0.05315585, -0.07466243, -0.01071489, …, -0.09742906,
         -0.02194229, -0.01105782],
        [-0.01084162,  0.13738561, -0.00613452, …,  0.06854109,
          0.00145186, -0.05325739],
        …,
        [-0.01722373,  0.07507852, -0.04060895, …, -0.02508039,
          0.03919973,  0.02877002],
        [-0.11121067,  0.03540494,  0.08640522, …, -0.06775279,
         -0.01658493, -0.01870682],
        [ 0.02905254,  0.04715664, -0.03811729, …, -0.05770208,
         -0.00215725,  0.01934869]]),
 'documents': ['Apple is preparing to launch the new iPhone 16 with AI-powered
features.',
  'Pineapple farming has increased in tropical countries due to rising demand.',
  'Cristiano Ronaldo scored a hat-trick in the UEFA Champions League match.',
  'NASA successfully landed a new rover on Mars to search for signs of life.',
  'The Indian government announced new policies for electric vehicle adoption.',
  'A breakthrough in cancer treatment shows promising results in clinical
trials.',
  'Tourists are flocking to Japan to see cherry blossoms this spring.',
  'Microsoft released a new update for Windows 12 with enhanced security.',
  'Climate change is causing rapid melting of glaciers in the Himalayas.',
  'A famous Bollywood actor announced his retirement after 30 years in film.'],
 'uris': None,
 'included': ['metadatas', 'documents', 'embeddings'],
 'data': None,
 'metadatas': [None, None, None, None, None, None, None, None, None, None]}
```

9 Insert Multiple Documents into the Collection

[2]:
```
import chromadb

#  Create a Chroma client (in-memory)
client = chromadb.Client()

#  Delete old collection if exists, then create a new one
client.delete_collection("news")
collection = client.create_collection(name="news")

#  Insert multiple diverse documents
collection.add(
    documents=[
        "Apple is preparing to launch the new iPhone 16 with AI-powered␣
 ↪features.",
        "Pineapple farming has increased in tropical countries due to rising␣
 ↪demand.",
        "Cristiano Ronaldo scored a hat-trick in the UEFA Champions League␣
 ↪match.",
        "NASA successfully landed a new rover on Mars to search for signs of␣
 ↪life.",
        "The Indian government announced new policies for electric vehicle␣
↪adoption.",
        "A breakthrough in cancer treatment shows promising results in clinical␣
↪trials.",
        "Tourists are flocking to Japan to see cherry blossoms this spring.",
        "Microsoft released a new update for Windows 12 with enhanced security.
↪",
        "Climate change is causing rapid melting of glaciers in the Himalayas.",
        "A famous Bollywood actor announced his retirement after 30 years in␣
↪film.",
        "Germany is investing heavily in renewable energy to reduce carbon␣
↪emissions.",
        "The World Cup final between Argentina and Brazil drew millions of␣
↪viewers.",
        "Scientists discovered a new species of bird in the Amazon rainforest.",
        "Stock markets surged today after positive economic growth data was␣
↪released.",
        "Researchers are working on quantum computers that can revolutionize AI.
↪",
    ],
    ids=[f"id{i}" for i in range(1, 16)]
)

#  Query the collection
results = collection.query(
    query_texts=["Latest updates on space exploration and Mars missions"],
    n_results=3
)

print(" Search Results for Space Exploration query:")
print(results)
```

```
 Search Results for Space Exploration query:
{'ids': [['id4', 'id8', 'id6']], 'embeddings': None, 'documents': [['NASA
successfully landed a new rover on Mars to search for signs of life.',
'Microsoft released a new update for Windows 12 with enhanced security.', 'A
breakthrough in cancer treatment shows promising results in clinical trials.']],
'uris': None, 'included': ['metadatas', 'documents', 'distances'], 'data': None,
'metadatas': [[None, None, None]], 'distances': [[0.8818884491920471,
1.4795515537261963, 1.545291781425476]]}
```

10 Using a Custom Embedding Function (Optional)

In ChromaDB, you can use a **custom embedding function** to generate vector representations of your documents.
One popular option is the **SentenceTransformerEmbeddingFunction** from the `sentence-transformers` library:

```
from chromadb.utils import embedding_functions

ef = embedding_functions.SentenceTransformerEmbeddingFunction(
    model_name="all-MiniLM-L6-v2"  # Lightweight and fast model
)
```

You can then pass this embedding function when creating a collection:

```
collection = client.create_collection(
    name="news_v2",
    embedding_function=ef
)
```

> Note: This requires the `sentence-transformers` package installed and compatible versions of `transformers`. On Windows, installation may require additional setup (Rust toolchain) for some versions. If you do not want to install extra packages, you can use ChromaDB's **default embedding function**, which works out-of-the-box.

11 Key Points to Remember

- Vector DBs = store embeddings + enable semantic search.
- Syntax differs, but concepts are **always the same: Add → Store → Query → Get Results**
- Don't memorize syntax → always check official docs.
- ChromaDB is great for **local testing & learning.** For large-scale apps → Pinecone, Weaviate, or Milvus.