SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 02 · RAG & Knowledge Systems

06

Vector Databases & Embeddings

Pinecone, Weaviate, pgvector — semantic search fundamentals

Module 06 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Pinecone, Weaviate, pgvector — semantic search fundamentals. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	5	06/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 02

02 Learning Objectives

- Explain embeddings and semantic (vs keyword) search.
- Compare vector database options and indexing methods.
- Build a semantic search index over your own data.
- Index multiple modalities in one store.

03 Core Concepts

Embeddings as meaning

An embedding model maps text (or images/audio) to a vector where semantic similarity equals geometric closeness. 'car' and 'automobile' land near each other even with no shared words — enabling search by meaning rather than keywords.

Similarity & indexing

Search ranks by cosine similarity / dot product. Brute force is O(n); approximate-nearest-neighbour indexes (HNSW, IVF) make billion-scale search fast with a small recall trade-off. Choose index parameters for your latency/recall budget.

Choosing a store

Pinecone and Weaviate are managed/feature-rich; pgvector adds vectors to Postgres so you keep one database; FAISS/Chroma are great for local prototypes. Pick based on scale, ops burden, and whether you need metadata filtering.

Multimodal indexing

With aligned encoders you can store text, image, and audio vectors together and query across them — find images by text, or related audio by an image — all from one index.

04 Visual Diagram

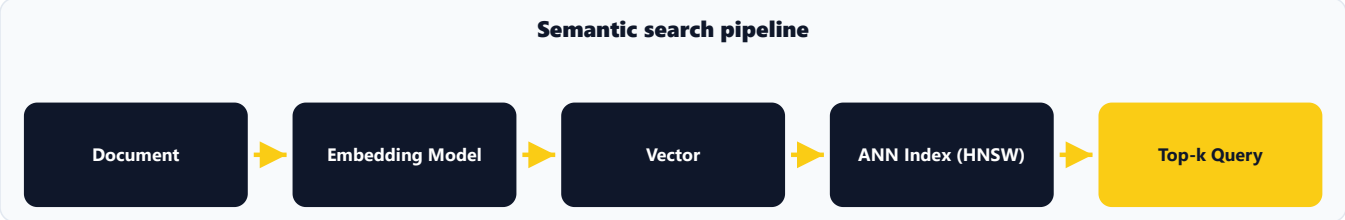

Figure 6 — Text becomes a vector; an approximate-nearest-neighbour index makes similarity search fast at scale.

05 Formula Spotlight

Cosine similarity

$$\cos(\theta) = (a \cdot b) / (\|a\| \cdot \|b\|)$$

a, b	two embedding vectors
$a \cdot b$	dot product of the vectors
$\|a\|, \|b\|$	their magnitudes (lengths)
$\cos(\theta)$	ranges −1…1; closer to 1 = more semantically similar

Semantic relevance is measured as the angle between embedding vectors. Because it ignores magnitude, cosine similarity compares meaning rather than length — the core ranking signal in every vector database.

06 Key Frameworks & Tools

Pinecone · Weaviate · pgvector (Postgres) · Chroma / FAISS · Sentence-Transformers

07 Hands-On Build

🛠 Multimodal Index (Image / Audio / Text) Vector Store

1. Pick an embedding model and embed a corpus of text documents.
2. Upsert vectors + metadata into a vector store (start with Chroma locally).
3. Run top-k semantic queries and inspect relevance.
4. Add image embeddings and query across modalities.

08 Code Walkthrough

Semantic search with Chroma — PYTHON

```python
import chromadb
from chromadb.utils import embedding_functions

ef = embedding_functions.SentenceTransformerEmbeddingFunction("all-MiniLM-L6-v2")
client = chromadb.Client()
col = client.create_collection("docs", embedding_function=ef)

col.add(documents=["Refunds take 5-7 business days.",
                   "Reset your password from the account page."],
        ids=["d1", "d2"])
print(col.query(query_texts=["How long for my money back?"], n_results=1))
```

09 🧪 Lab Assistant

Lab 6 · Build a local semantic index

Prerequisites:pip install chromadb sentence-transformers

1. **Add a few documents to a Chroma collection**
   > Expected: Documents stored with auto-generated embeddings
2. **Query with a paraphrase (different words, same meaning)**
   > Expected: The semantically matching doc ranks first
3. **Add a metadata filter to the query**
   > Expected: Results narrowed to the matching subset

Troubleshooting

- Irrelevant results → ensure the same embedding model for add and query.
- Slow at scale → switch to a server-backed store (pgvector/Pinecone).

10 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Mixing embedding models between indexing and querying.
- Ignoring metadata filtering, then over-relying on vector recall.
- Choosing exact search at scale where ANN is needed.

✓ KEY TAKEAWAYS

- Embeddings turn meaning into geometry for semantic search.
- ANN indexes (HNSW/IVF) make large-scale retrieval fast.
- pgvector keeps everything in one database; Pinecone/Weaviate scale managed.

📺 **Companion video:** Watch the module 06 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

11 Further Resources

pgvector docs • Pinecone & Weaviate quickstarts • MTEB embedding leaderboard