Modern Embeddings: A Practical Guide

How to Use Embeddings in 2025 with Hugging Face, RAG, and Vector Databases

Prof. Dr. Jörg Osterrieder

Methods and Algorithms — MSc Data Science

Spring 2026

Where We Are Going

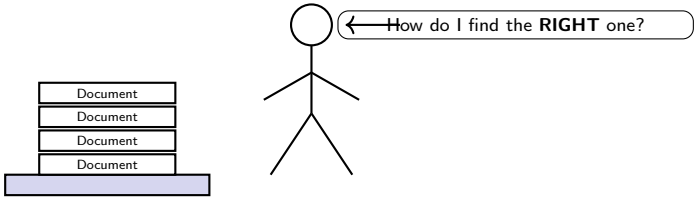
XKCD #1838 by Randall Munroe (CC BY-NC 2.5)

You already know **what embeddings are** (L06a).

Now: how do practitioners actually **USE** them today?

- Sentence-level embeddings
- Semantic search and RAG
- Vector databases

The Problem: So Many Documents!



Modern embeddings solve this: they let you **search by meaning**, not keywords.

Keyword search fails when the user says "revenue" but the document says "income".

From Words to Sentences

Word Embeddings (L06a)

- One vector per **word**
- "bank" always the same vector
- Word2Vec, GloVe

Sentence Embeddings (Today)

- One vector per **sentence**
- Captures full meaning
- sentence-transformers

Key Insight

Modern applications embed **sentences**, not individual words.

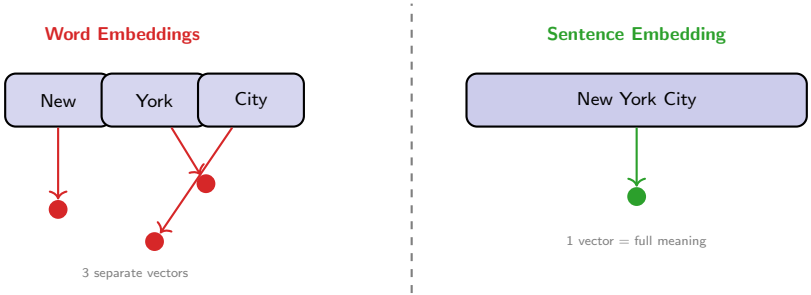
sentence-transformers is the most popular library: 100M+ downloads on Hugging Face.

The Key Players in 2025

- **Open source:** sentence-transformers
  Models: `all-MiniLM-L6-v2`, `all-mpnet-base-v2`
- **API-based:** OpenAI (`text-embedding-3-small`), Cohere (`embed-v3`), Voyage AI
- **Leaderboard:** MTEB on Hugging Face ranks hundreds of models by task

Tip: start with all-MiniLM-L6-v2 (free, fast, good). Upgrade to API models if you need top accuracy.

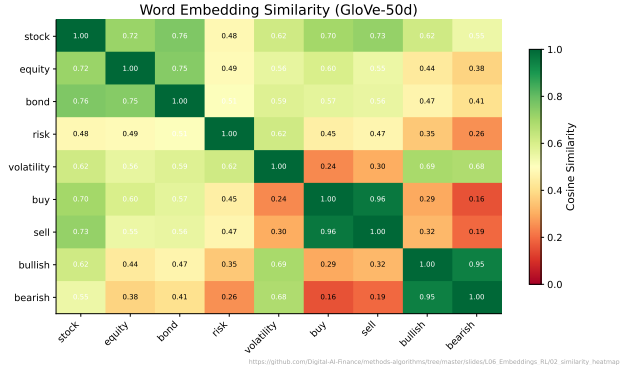
Sentence vs. Word Embedding



Sentence embeddings understand that "New York City" is **one concept**, not three words.

Technical detail: sentence-transformers use mean pooling over token embeddings from a transformer model.

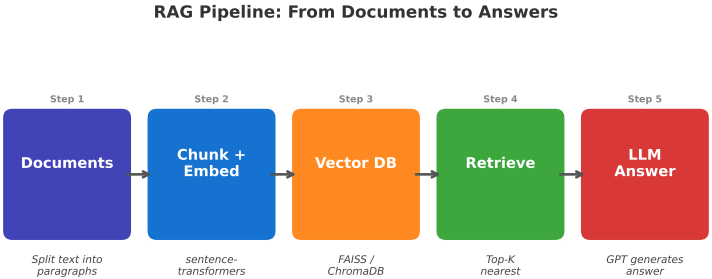
Semantic Search = Cosine Similarity



- This is the **core operation** behind semantic search
- Compute cosine similarity between query and document embeddings
- Darkest cell = most relevant result

Cosine similarity ranges from -1 (opposite) to $+1$ (identical). In practice, scores above 0.7 are strong matches.

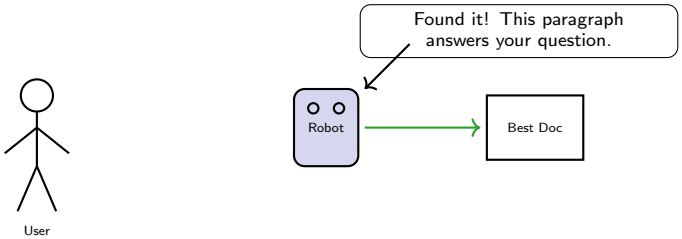
RAG: The #1 Pattern for Embeddings + LLMs



- **RAG** = Retrieval-Augmented Generation
- Embed your documents once, retrieve on every question
- No retraining needed — works with any LLM

RAG lets an LLM answer questions about YOUR documents without retraining. Used at JPMorgan, Bloomberg, every major bank.

The Smart Librarian: How RAG Works



1. Embed documents once, store vectors in a database (FAISS, ChromaDB, Pinecone)
2. When a question arrives, embed it and find the nearest documents
3. Feed those documents to an LLM for a precise answer

Vector databases: FAISS (Meta, free), ChromaDB (open source), Pinecone (cloud). All work with sentence-transformers.

Try It Yourself — Six Lines to Semantic Search

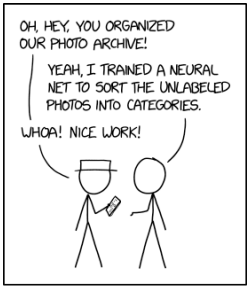
```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer("all-MiniLM-L6-v2")

docs = ["Annual revenue rose 12%",
        "Q3 losses exceeded forecast"]
query = "How did the company perform financially?"

doc_embeddings = model.encode(docs)
query_embedding = model.encode(query)
# Use cosine similarity to find the best match!
```

That's it — **six lines** to build a semantic search engine.

---

pip install sentence-transformers. Free, runs locally, no API key needed. Hugging Face Hub has 5000+ embedding models.

What Could Possibly Go Wrong?



XKCD #2173 by Randall Munroe (CC BY-NC 2.5). Next: try the RAG notebook with real financial filings!