

Material : **Generative AI**

Topic : **Vector Database**



Daniel
danielgenai77@gmail.com

Gen AI – Vector Database

1. Data	2
2. Types of data	2
2.1. Structured Data:	2
2.2. Semi-structured Data:	2
2.3. Unstructured Data:	2
3. Why Vector Database?	2
4. Embedding	3
4.1. Why Embeddings?	3
4.2. Example: Word Embeddings	3
4.3. Types of Data That Can Be Embedded	4
4.4. Real-World Uses	4
5. Before embedding	4
6. Vector Database	5
7. Open-Source Models for Embeddings	5
8. Environment setup: sentence-transformers	6
9. Convert Text to Embeddings using transformers	6
10. Convert Text to Embeddings using transformers	7
11. Clustering of Similar Concepts in Word Embeddings	9
12. Semantic Search with Embeddings and Vector Databases	9
13. Why Encode Unstructured Data?	10
14. Examples	10
14.1. Example #1: Vacation Photos	10
14.2. Example #2: News Search	13
15. How to generate embeddings?	14
16. How Word Embeddings Were Created Before Transformers	14
17. Popular Pre-Transformer Embedding Models	15
18. BERT & Sentence Embeddings	15
19. Vector Database Providers	15

Gen AI – Vector Database

1. Data

- ✓ Data is a collection of facts or information used for analysis and decision-making.

2. Types of data

- ✓ There are mainly three types of data.
 - Structured data
 - Semi structured data
 - Unstructured data

2.1. Structured Data:

- ✓ Organized in clear rows and columns, like spreadsheets or databases (e.g., names, dates).

2.2. Semi-structured Data:

- ✓ Partially organized but doesn't fit rigid tables.
- ✓ It uses tags or markers to separate data elements, like JSON or XML files.

2.3. Unstructured Data:

- ✓ No fixed format, including text, images, videos, and audio.

3. Why Vector Database?

- ✓ Unstructured data includes information that doesn't fit into rows and columns, such as text, images, audio, and video.
- ✓ Because this data lacks a fixed format.
- ✓ Traditional databases struggle to store and search it efficiently.
- ✓ Vector databases solve this problem.

4. Embedding

- ✓ Embeddings are numerical representations of data (like text, images, or audio) in a way that captures its meaning, context, or key features.

4.1. Why Embeddings?

- ✓ Real-world data is messy — documents, images, videos, audio — and machines can't "understand" them directly.
- ✓ Embeddings translate this data into a numerical form that preserves its meaning, semantic relationships, or structure.

4.2. Example: Word Embeddings

- ✓ Words like:
 - "king" → [0.25, 0.11, -0.78, ...]
 - "queen" → [0.21, 0.09, -0.80, ...]
 - "apple" → [-0.44, 0.87, 0.31, ...]
- ✓ You'll notice that:
 - king - man + woman \approx queen → embeddings capture relationships!
 - "king" and "queen" are closer than "king" and "apple" in vector space.


4.3. Types of Data That Can Be Embedded

Data Type	Embedding Example
✓ Text (word/sentence)	✓ Word2Vec, BERT, GPT embeddings
✓ Images	✓ CLIP, ResNet embeddings
✓ Audio	✓ Whisper, Wav2Vec embeddings
✓ Documents	✓ Sentence Transformers, Doc2Vec

4.4. Real-World Uses

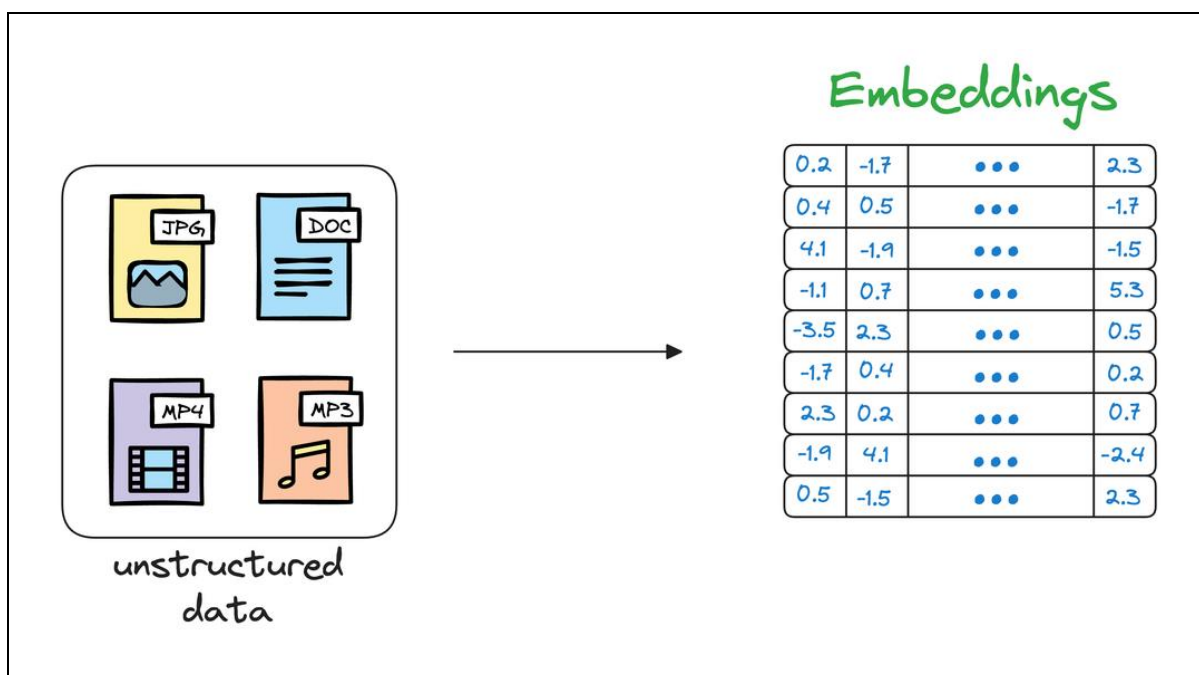
- ✓ **Search engines:** Retrieve semantically similar documents.
- ✓ **Recommendation systems:** Recommend items based on similarity.
- ✓ **Chatbots/LLMs:** Understand user inputs more deeply.
- ✓ **Clustering/classification:** Group similar items together.

5. Before embedding

 Why Embeddings Replaced These		
Feature	One-Hot / BoW / TF-IDF	Embeddings
Understand meaning?	✗	✓
Dense vector?	✗ Sparse	✓ Dense
Captures similarity?	✗	✓
Learns from context?	✗	✓ (especially with neural models)

6. Vector Database

- ✓ Vector database stores unstructured data (text, images, audio, video, etc.) in the form of vector embeddings (numeric representations).
- ✓ Each data point (e.g., word, image, document) is converted into a numerical vector called an *embedding*.



7. Open-Source Models for Embeddings

- ✓ All these are available via Hugging Face.

Model Name	Description
all-MiniLM-L6-v2	suitable for semantic search
multi-qa-MiniLM-L6-cos-v1	Optimized for question answer tasks
paraphrase-MiniLM-L12-v2	Great for comparing similar sentences

8. Environment setup: sentence-transformers

```
pip install sentence-transformers
```

9. Convert Text to Embeddings using transformers

Program Name	<pre>Convert Text to Embeddings using transformers demo1.py from sentence_transformers import SentenceTransformer # Load a pre-trained embedding model model = SentenceTransformer('all-MiniLM-L6-v2') # Your input text text = "The Eiffel Tower is located in Paris and was built in 1889." # Get the embedding embedding = model.encode(text) # Print part of the embedding print("Embedding vector (truncated):", embedding[:10])</pre>
Output	<pre>Embedding vector (truncated): [3.9757401e-02 5.1625822e-02 1.9713903e-05 1.6055735e-03 1.1941780e-02 -1.3450466e-02 -8.6841680e-02 3.2891795e-02 5.2027605e-03 -3.1023417e-02]</pre>

10. Convert Text to Embeddings using transformers

```
Program Name    Multiple Sentence Embeddings Example
demo1.py

from sentence_transformers import SentenceTransformer
import numpy as np

# Load a pre-trained embedding model
model = SentenceTransformer('all-MiniLM-L6-v2')

# List of sentences to embed
sentences = [
    "The Eiffel Tower is located in Paris and was built in 1889.",
    "Paris is home to the famous Eiffel Tower.",
    "The Statue of Liberty is in New York City.",
    "Bananas are yellow and rich in potassium.",
    "Artificial intelligence is transforming the world."
]

# Generate embeddings for each sentence
embeddings = model.encode(sentences)

# Print the first 10 values of each embedding
for i, (sentence, embedding) in enumerate(zip(sentences,
embeddings)):
    print("Sentence {i+1}: {sentence}")
    print("Embedding vector (truncated):", embedding[:10])
```

Output

```
Sentence 1: The Eiffel Tower is located in Paris and was built in
1889.
Embedding vector (truncated): [ 3.97573858e-02  5.16258739e-
02  1.97258523e-05  1.60556904e-03
 1.19417915e-02 -1.34504791e-02 -8.68416578e-02
 3.28918062e-02
 5.20278560e-03 -3.10233738e-02]
```


Sentence 2: Paris is home to the famous Eiffel Tower.

Embedding vector (truncated): [0.08414423 0.04866021
0.01386443 0.00248201 0.04932949 0.00215907
-0.0550714 0.01174218 -0.0026298 -0.02511607]

Sentence 3: The Statue of Liberty is in New York City.

Embedding vector (truncated): [0.08450228 0.09561136
0.03509006 0.01049995 0.04243909 0.01134018
-0.00709057 -0.0134034 0.00454255 -0.01797276]

Sentence 4: Bananas are yellow and rich in potassium.

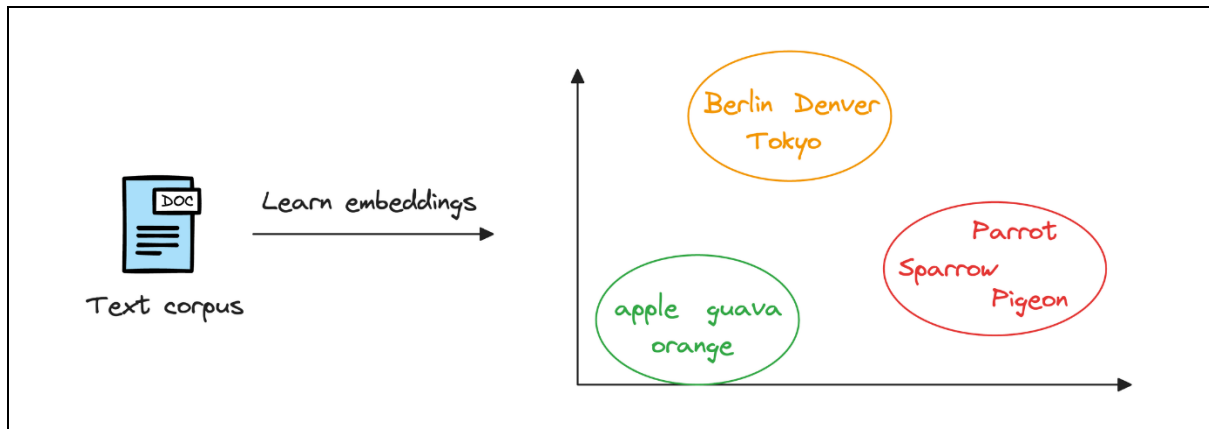
Embedding vector (truncated): [-0.00945398 -0.03126404
0.02787722 0.03488883 0.00328189 0.07226413
0.0417771 -0.01045569 0.02283465 0.07436628]

Sentence 5: Artificial intelligence is transforming the world.

Embedding vector (truncated): [0.03872412 -0.00110552
0.08271616 -0.01628856 0.04654312 -0.0095303
-0.02997492 0.00349416 0.01119627 0.00263023]

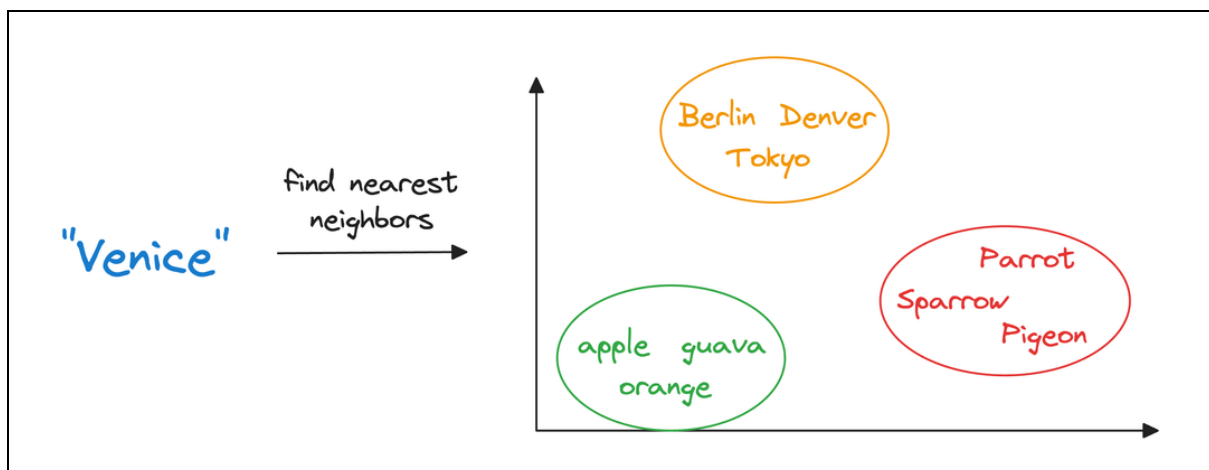
11. Clustering of Similar Concepts in Word Embeddings

- ✓ In word embeddings, similar concepts—like fruits or cities—tend to cluster together in the embedding space.



12. Semantic Search with Embeddings and Vector Databases

- ✓ This shows that, when properly trained, embeddings can capture the semantic meaning of the entities they represent.
- ✓ Once stored in a vector database, embeddings allow us to retrieve original objects similar to a given query on unstructured data.



13. Why Encode Unstructured Data?

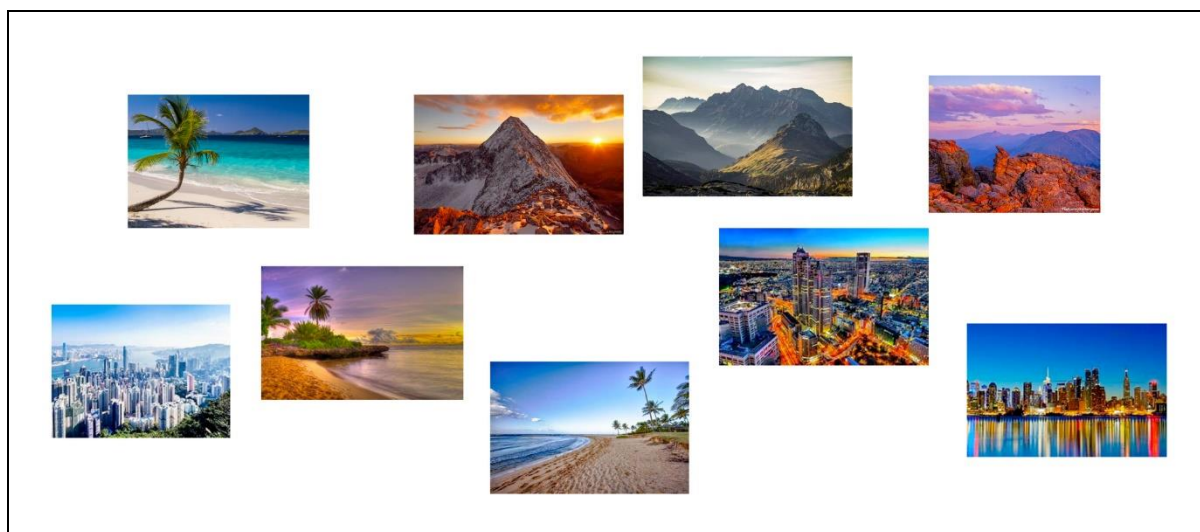
- ✓ Encoding unstructured data makes it easier to perform tasks like search, clustering, and classification.
- ✓ For example, when an e-commerce site recommends similar products, it's often powered by vector databases.

14. Examples

- ✓ **Image Search:** Find similar images by comparing vector representations.
- ✓ **Product Recommendations:** Suggest related items using product vectors.
- ✓ **Semantic Text Search:** Match user queries with relevant results based on meaning, not just keywords.

14.1. Example #1: Vacation Photos

- ✓ Imagine a collection of vacation photos—beaches, mountains, cities, forests.
- ✓ By converting these images into vectors, we can easily find and group similar scenes, like all beach photos, without manual tagging.



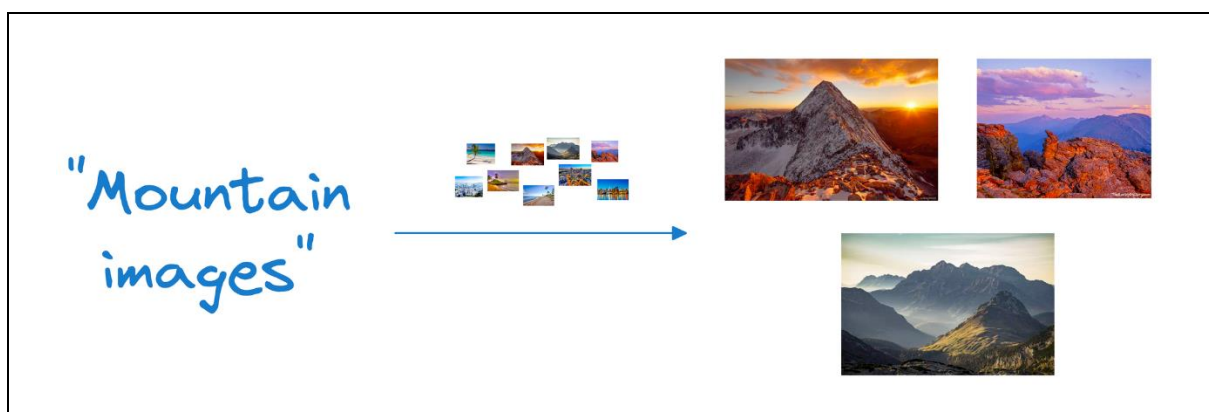
The Need for Smarter Organization

- ✓ Now, we want to organize these photos to easily find similar ones.
- ✓ We can sort photos by date or location.
- ✓ But that doesn't help if we want all beach photos or city



Smarter Search with Vectors

- ✓ We convert each photo into a vector based on features like color, shape, and texture.
- ✓ These vectors sit in a multi-dimensional space.
- ✓ To find similar photos—like mountains—we encode a text query into a vector and retrieve the closest image vectors.
- ✓ This makes searching by content fast and intuitive.



Important Note

- ✓ A vector database is not just for storing embeddings—it also keeps the original data (like images or text) linked to those embeddings, enabling efficient retrieval and meaningful results.

Why Store Raw Data Too?

- ✓ If the database only stores vectors, we can't show actual results.
- ✓ For image search, users need the images—not just vectors—so the database must store both to return meaningful results.

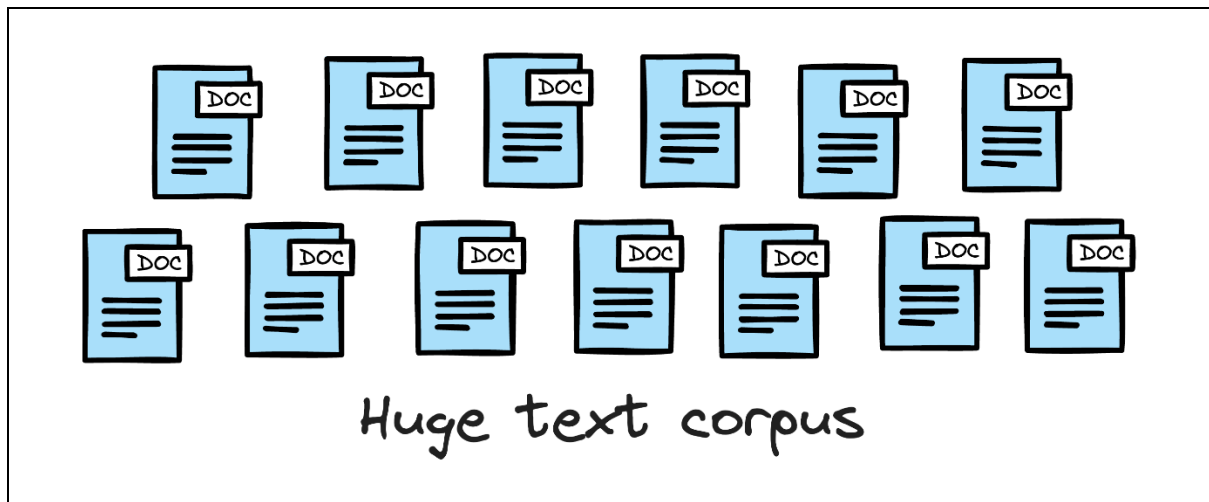


Complete Retrieval

- ✓ By storing both embeddings and raw data, a vector database ensures that search results include not just similar vectors but also the actual images users want to see.

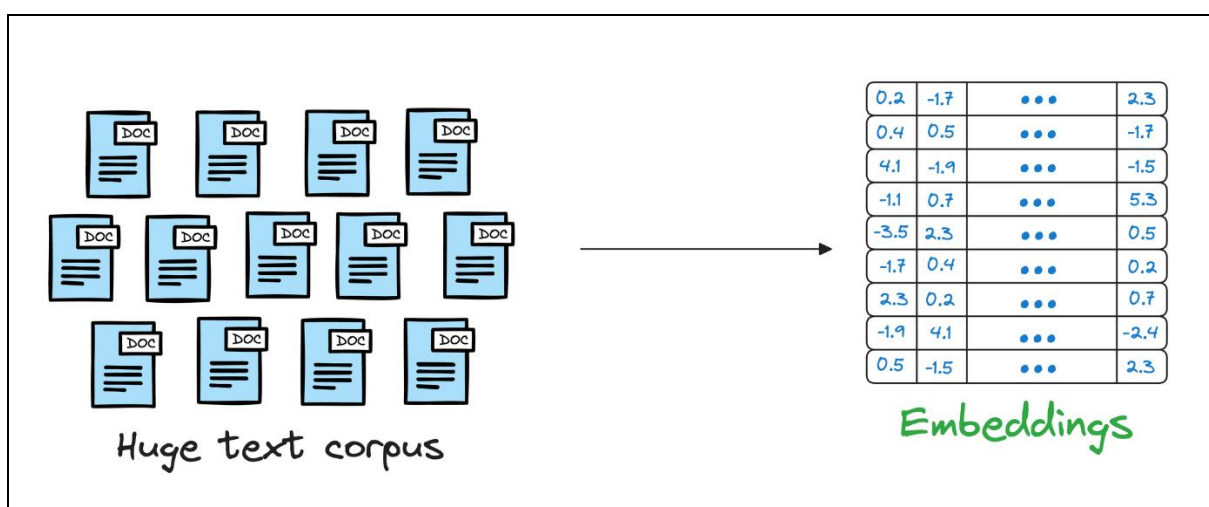
14.2. Example #2: News Search

- ✓ With thousands of news articles, vector search helps find relevant answers by matching meaning, not just keywords—making retrieval more accurate and insightful.



Limitations of Keyword Search

- ✓ Traditional keyword search misses the nuance of language—different phrases can mean the same thing.
- ✓ By encoding text as vectors, we capture meaning, enabling smarter and more flexible search in a vector database.



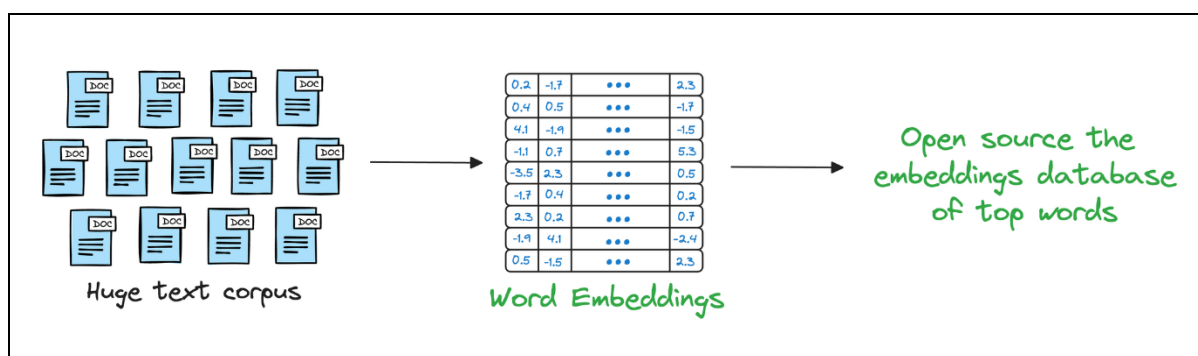
Smarter Matching

- ✓ A vector database compares the query's vector with text vectors, allowing it to find relevant results—even when the wording is different.

15. How to generate embeddings?

- ✓ Embeddings change words or sentences into numbers.
- ✓ So, computers can understand their meaning.
- ✓ This helps with things like search, matching, or sorting text.

16. How Word Embeddings Were Created Before Transformers



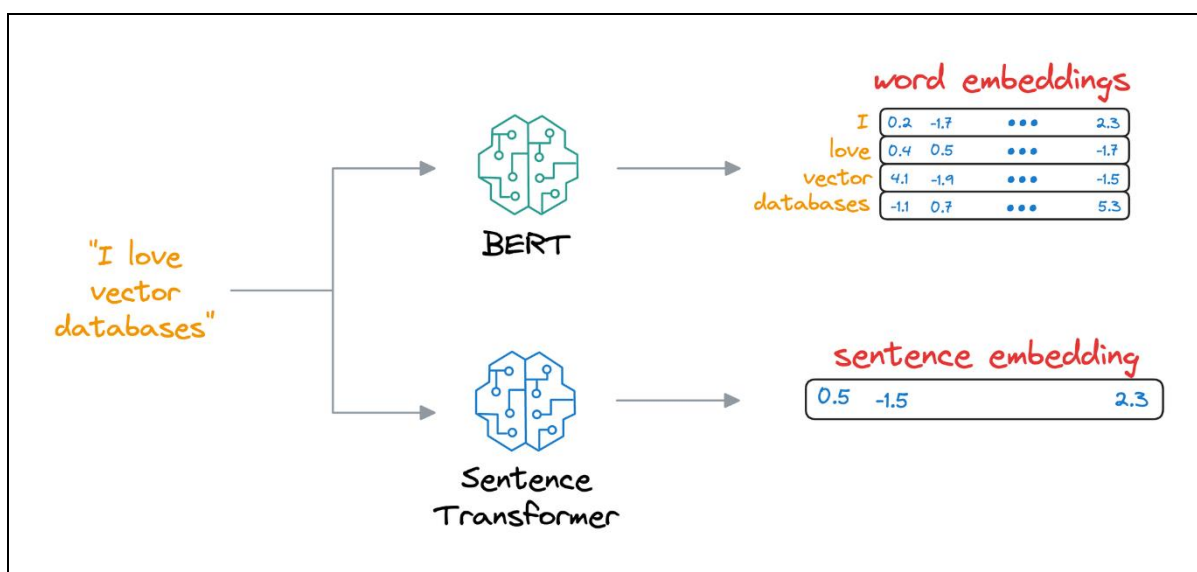
- ✓ This image shows how word embeddings were built before Transformers:
 - **Huge text corpus:** A large collection of documents is used as training data.
 - **Word embeddings:** Words from the text are converted into numerical vectors using deep learning.
 - **Open-source database:** The resulting embeddings for common words are shared publicly for reuse.
- ✓ In short:
 - Train on lots of text → get word vectors → share for others to use.

17. Popular Pre-Transformer Embedding Models

- ✓ Others used these shared embeddings in their projects.
- ✓ Models like **Word2Vec**, **GloVe**, and **FastText** (2013–2017) were widely used and showed strong results in capturing word relationships.

18. BERT & Sentence Embeddings

- ✓ **BERT**: A language model trained using two techniques:
 - **Masked Language Modeling (MLM)**: Predict a missing word in the sentence, given the surrounding words.
 - **Next Sentence Prediction (NSP)**.
- ✓ **SentenceTransformer**: Essentially, the SentenceTransformer model takes an entire sentence and generates an embedding for that sentence.



19. Vector Database Providers

- ✓ **Pinecone** : Managed, fast, and scalable vector search.
- ✓ **Weaviate** : Open-source, ML-powered, supports multiple data types.
- ✓ **Milvus** : Open-source, built for large-scale similarity search.
- ✓ **Qdrant** : Filter-rich, production-ready semantic search engine.