# 📘 AI Notes – Understanding Embeddings

These are your study notes on embeddings – one of the most important building blocks of AI systems like RAG, knowledge base agents, and semantic search. This document is written in a clear, simple, and elegant way so you can understand and never forget the concepts.

## 1. What Are Embeddings?

An embedding is a list of numbers (called a vector) that represents the meaning of a piece of text. Think of it as a unique fingerprint of meaning. Instead of computers working with plain English words, embeddings turn text into mathematical objects so computers can understand and compare meanings.

Example:

"car insurance" → [0.12, -0.88, 0.53, …]

"auto policy" → [0.13, -0.87, 0.52, …]

"banana smoothie"→ [-0.45, 0.12, -0.66, …]

In this list, each number is a coordinate. Together, they form a point in a giant 'semantic space'. Similar sentences sit close together, while unrelated ones sit far apart.

## 2. What Does 'Semantic' Mean?

Semantic means 'meaning'. It is different from syntax, which is about the exact words. For example, 'car insurance' and 'auto policy' are different syntactically, but semantically they mean the same thing. Embeddings capture semantic meaning, so we can build search systems that understand concepts, not just words.

## 3. Why Do We Convert Text into Numbers?

Computers don’t understand English. They only understand numbers. If we keep text as plain English, we can only do exact keyword matches. That means 'house coverage' may never match 'homeowners insurance'.

By converting text into embeddings (numbers), we can calculate distances between meanings. Close vectors = similar meaning. Far vectors = different meaning.

Analogy:

Text in English is like the name of a place ('Eiffel Tower'). Embeddings are like GPS coordinates (48.8584, 2.2945). With just names, you can’t measure distance. With coordinates, you can.

## 4. Do Embeddings Map Into Space?

Yes! Every embedding is a coordinate in a huge multi-dimensional space. For OpenAI’s models like 'text-embedding-3-small', each embedding has 1536 numbers. That means each sentence becomes a point in a 1536-dimensional space.

In that space, similar meanings are close together. 'car insurance' and 'auto policy' land near each other. 'banana smoothie' is far away.

## 5. Why 1536 Numbers?

The 1536 numbers are the 'resolution of meaning'. Just like an image can be low-resolution (128 pixels) or high-resolution (4K), embeddings have a certain number of dimensions. 1536 is a balance chosen by the model designers – it’s rich enough to capture detailed meaning, while still being efficient to store and search.

Think of each number as one feature describing the text – like color, shape, size for objects. Together, 1536 features describe the semantic fingerprint of a sentence.

## 6. Insurance Example

Let’s embed three rules from insurance documents:  
1. 'HO-3 excludes sewer backup unless endorsed.'  
2. 'Water backup is covered with endorsement form HO-234.'  
3. 'Flood damage is excluded under homeowners policies.'  
  
Now, if a user asks: 'Does this policy cover water discharge?', the embedding will land close to rules 1 and 2 (relevant) but far from rule 3 (not relevant). This is why embeddings are so powerful in insurance knowledge bases.

## 7. Key Takeaways

• Embedding = fingerprint of meaning.

• Sentence → embedding (vector of numbers).

• Similar meanings → vectors close together in space.

• pgvector in Postgres stores embeddings so we can search them.

• In insurance, this lets us build smart knowledge agents that understand concepts.

📌 Remember: embeddings are the foundation of modern AI search and reasoning. They turn text into coordinates in a space of meaning. That’s how AI can find, compare, and understand information.

**8. How We Compare Embeddings**

**We need a way to measure “closeness”:**

* **Cosine similarity → angle between vectors (most common).**
* **Euclidean distance → straight-line distance.**
* **Dot product → how aligned they are.**

**Cosine similarity gives a value between -1 and 1.**

* **1 = same direction (identical meaning).**
* **0 = orthogonal (unrelated).**
* **-1 = opposite meaning.**
* Example with short vectors:
* Query: [1, 2]
* Chunk A: [1, 2] → very close
* Chunk B: [-2, -1] → far (opposite direction)