What are Embeddings?

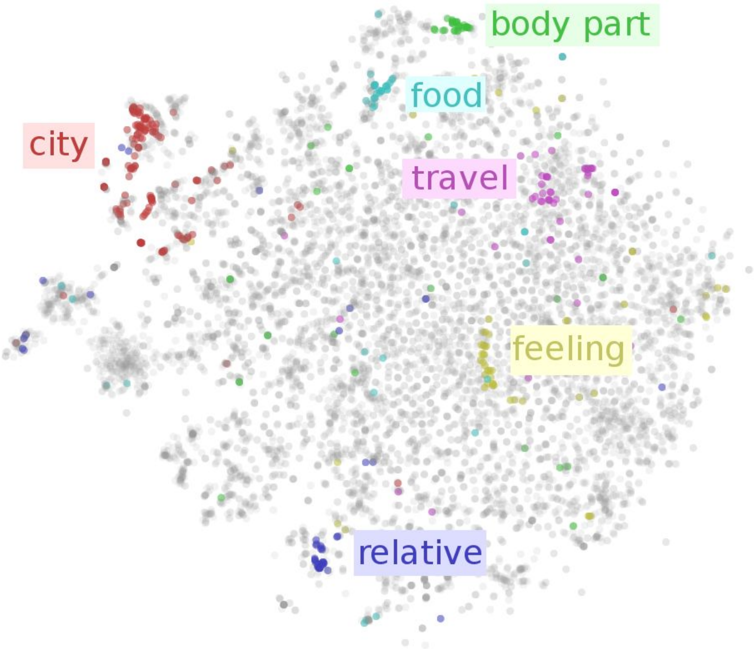
Learn word embeddings, how to use them and their various use cases

What are Embeddings?

Embeddings are numerical representations of data (text, images, audio) in a **high-dimensional vector space.**

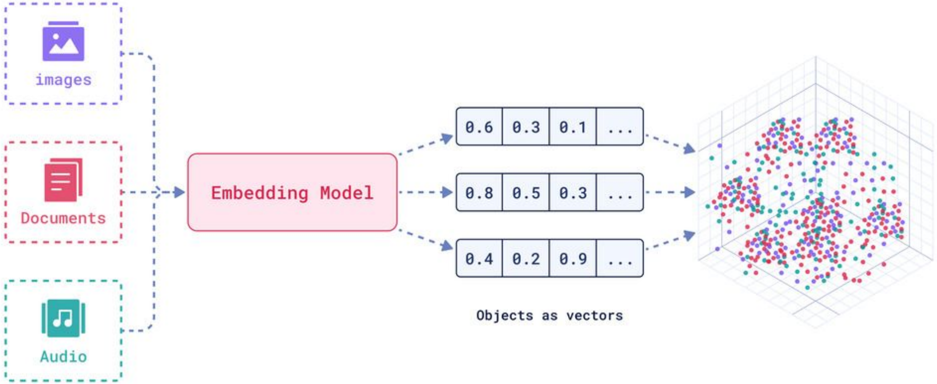
They capture **semantic meaning**, making **similar concepts appear close together** mathematically.

Core technology enabling many AI applications, especially in natural language processing.

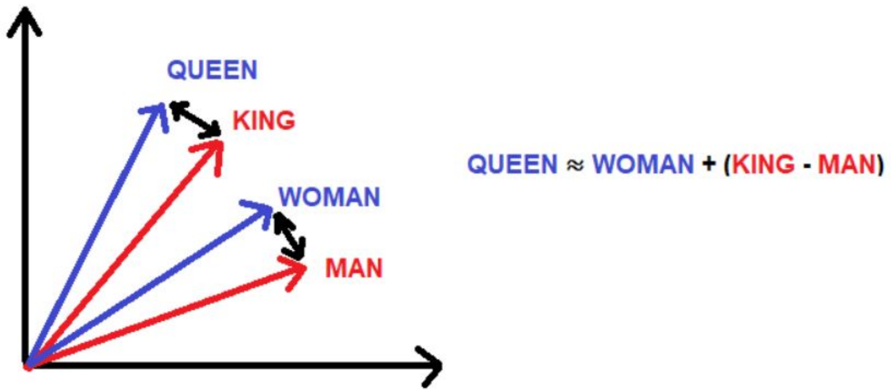


How Embeddings Work

1. **Tokenization:** Break text into tokens (words, subwords)
2. **Vector Creation:** Convert tokens/images into high-dimensional vectors
3. **Contextual Understanding:** Position vectors based on meaning in context



How Embeddings Work



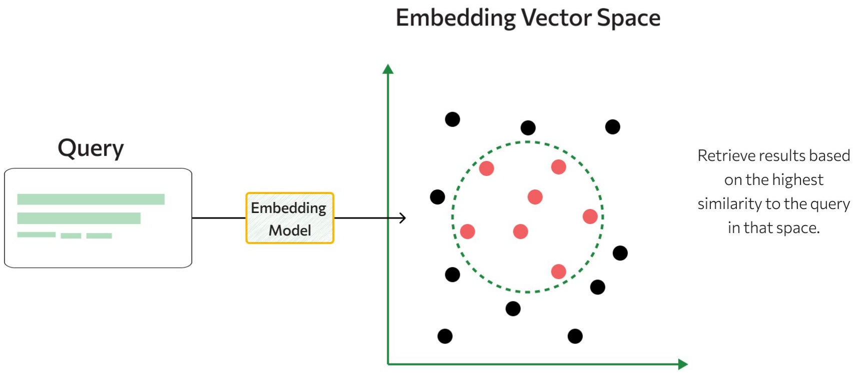
As embeddings represent a **space within multiple dimensions**, different embeddings will either **be closer or further away from other embeddings.**

Common Embedding Models

- **OpenAI:** text-embedding-ada-002, text-embedding-3-small/large
- **Cohere:** embed-english-v3.0, embed-multilingual
- **Sentence Transformers:** all-MiniLM-L6-v2, all-mpnet-base-v2
- **Google:** PaLM, BERT, Universal Sentence Encoder

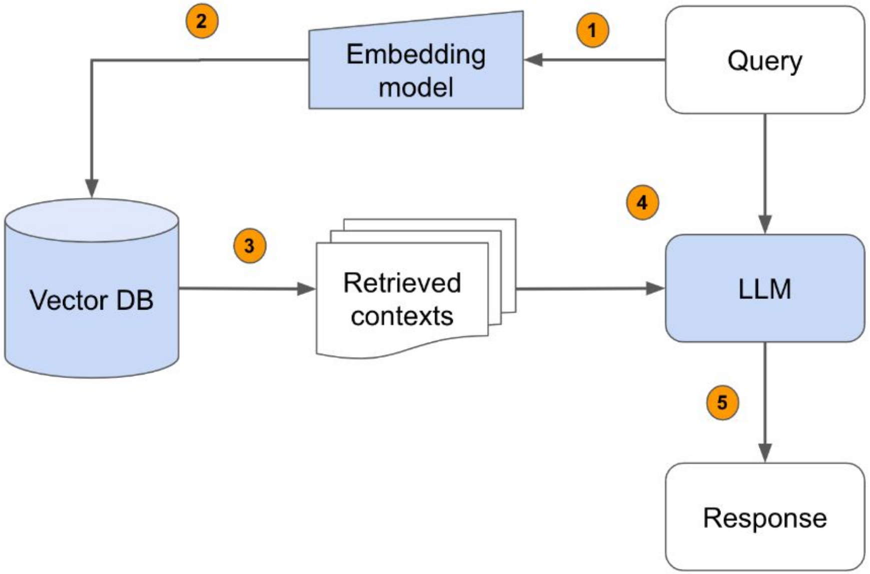
Applications in Prompt Engineering - Semantic Search

- Match queries to documents based on meaning, not just keywords
- Enable more intuitive information retrieval



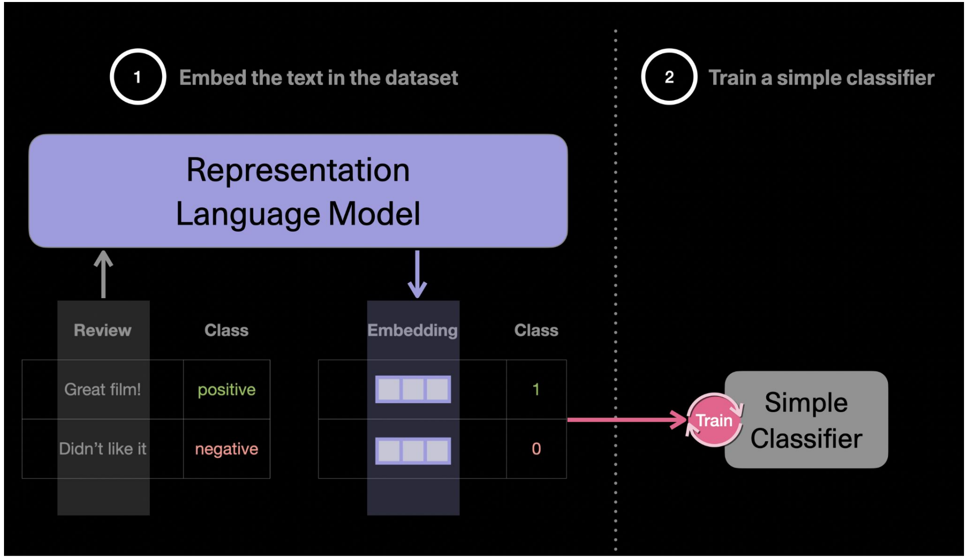
Applications - Retrieval-Augmented Generation (RAG)

- Find relevant information from a knowledge base
- Incorporate into prompts/chat history to provide context for LLMs



Applications in Prompt Engineering - Text Classification

- Categorize content based on semantic patterns
- Power content recommendation systems



Embeddings in Practice

```python
from openai import OpenAI
client = OpenAI()

response = client.embeddings.create(
    input="Your text string goes here",
    model="text-embedding-3-small"
)

print(response.data[0].embedding)
```



```json
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [
        -0.006929283495992422,
        -0.005336422007530928,
        -4.547132266452536e-05,
        -0.024047505110502243
      ],
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
  }
}
```

Measuring Similarity

- **Cosine similarity:** measures angle between vectors
- **Euclidean distance:** measures straight-line distance
- **Dot product:** simple multiplication of vectors

Cosine Similarity - The Heart of Embedding Comparisons

- Mathematical measure of similarity between two vectors regardless of their magnitude
- Calculates the cosine of the angle between vectors in a multi-dimensional space
- Ranges from -1 (opposite direction) to 1 (same direction), with 0 indicating orthogonality

Cosine Distance:

$$d_{cos}(x,y) = 1 - \frac{\sum_{i=0}^{n-1} x_i y_i}{\sqrt{\sum_{i=0}^{n-1} x_i^2} \times \sqrt{\sum_{i=0}^{n-1} y_i^2}}$$

Cosine Similarity - Coding Example

```python
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Two embedding vectors
embedding1 = np.array([0.2, 0.5, 0.3, 0.8, 0.1])
embedding2 = np.array([0.3, 0.4, 0.2, 0.7, 0.2])

# Calculate similarity
similarity = cosine_similarity([embedding1], [embedding2])[0][0]
print(f"Similarity score: {similarity:.4f}")  # Output: Similarity score: 0.9819
```

Advanced Techniques & Future Directions

- **Domain Specialization:** Fine-tuned embeddings for industry-specific language and applications
- **Multimodal Integration:** Models like CLIP and ImageBind creating unified vector spaces across text, images, and audio
- **Hybrid Approaches:** Combining neural embeddings with traditional methods (BM25/TF-IDF) for optimal results
- **Key Challenge:** Keeping embeddings updated with new knowledge without complete retraining

Next Steps 🚀

Let’s get hands on and practice learning how to create embeddings within OpenAI, and using cosine similarity to compare different embeddings!