Vector & Embeddings

by Jason Guillauto

Embeddings in Practice

Effectively implementing embeddings requires two key components: a robust embedding strategy and an efficient vector database.

Embedding Strategies

Embeddings transform data into high-dimensional vector spaces where semantic relationships are preserved as geometric relationships.

Core Embedding Models

- Word2Vec: Pioneered neural word representations (2013)
- GloVe: Global vectors capturing word co-occurrence statistics
- SWIVEL: Submatrix-wise vector embedding learner
- BERT: Bidirectional encoder transformers that revolutionized contextual embeddings

Training Approaches

- **Continuous Bag of Words:** Predicts a target word from surrounding context words by averaging their embeddings
- **Skip-Gram:** Uses a center word to predict surrounding context words, excelling with rare words

Types of Embeddings

- Multimodal embeddings: Represent different data types in shared space
- Structured data embeddings: Created by ML models for tabular data
- User/item embeddings: For recommendation systems
- Graph embeddings: Represent network relationships

Advanced Training Techniques

- Bidirectional deep neural networks for context awareness
- Unsupervised pre-training on unlabeled text corpora
- Subword tokenization to handle out-of-vocabulary words
- Contrastive learning to differentiate similar vs. dissimilar items

Evaluation Framework

- **Precision:** Percentage of retrieved items that are relevant
- **Recall:** Percentage of relevant items that are retrieved
- **NDCG:** Normalized Discounted Cumulative Gain measures ranking quality

Vector Databases

Vector databases provide efficient storage and retrieval of embeddings using similarity search algorithms.

Vector Search Fundamentals

Enables search on any type of data by comparing vector positions using:

- **Euclidean Distance:** Measures direct spatial distance between vectors
- **Cosine Similarity:** Compares the angles between vectors
- **Inner Product:** Dot product calculation for similarity

Approximate Nearest Neighbor (ANN) Techniques

Methods that optimize vector search for large datasets:

- **Locality Sensitive Hashing:** Hashes similar items to the same buckets
- **Tree-Based Methods:** KD-trees and Ball Trees for spatial partitioning
- **HNSW:** Hierarchical Navigable Small Worlds for efficient graph-based search
- **ScaNN:** Scalable Nearest Neighbors for fast, efficient vector search

Vector Database Implementations

Specialized databases optimized for vector storage and retrieval:

- Pinecone, Weaviate, Milvus: Purpose-built vector databases
- Qdrant, Faiss: Optimized for high-performance similarity search
- ChromaDB, LanceDB: Embedding-specific databases
- PostgreSQL + pgvector: SQL databases with vector extensions

Retrieval Augmented Generation (RAG)

Enhancing LLM responses with relevant context from vector databases:

1. Choose an embedding strategy appropriate for your data
2. Select a vector database with ANN capabilities
3. Store embeddings of knowledge base
4. Retrieve relevant context based on query similarity
5. Augment prompts with retrieved information