Introduction to Embeddings

From Concepts to Applications in AI Systems

July 5, 2025

Outline

Definition of Embeddings

Core Definition

An **embedding** is a representation of objects (such as numbers, words, sentences, or images) in a continuous vector space.

- Maps symbolic data (words, categories) to numerical vectors
- Semantic relationships captured by distances and directions
- Preserves relationships like similarity, context, and associations
- Fixed dimensionality, much smaller than original input

Key Insight

Embeddings translate complex, categorical inputs into meaningful numerical representations.

Types of Embeddings

1. Word Embeddings

- Examples: Word2Vec, GloVe
- Similar words have similar vectors
- "king" and "queen" are close

2. Sentence/Document Embeddings

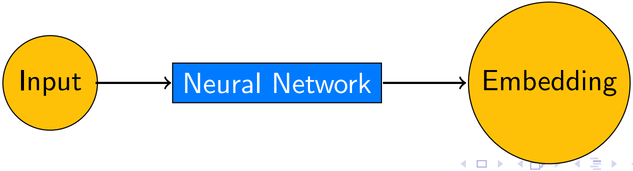
- Examples: Sentence-BERT, Universal Sentence Encoder
- Represent entire text chunks

3. Feature Embeddings

- For recommendation systems
- User IDs, product categories

4. Image Embeddings

- CNN feature maps
- Continuous image representations



Advantages of Embeddings

1. **Semantic Similarity**
    - Similar objects have closer vectors
    - Synonyms like "happy" and "joyful" cluster together
2. **Dimensionality Reduction**
    - Convert sparse inputs (one-hot encoded) to dense vectors
    - From 10,000 dimensions to 300 dimensions
3. **Computational Efficiency**
    - Continuous vectors are faster to compute
    - Better memory utilization
4. **Generalization**
    - Capture rich contextual features
    - Enable models to handle unseen examples

One-Hot vs Embeddings Example

One-Hot Encoding

$$\text{“cat”} = [0, 0, 0, 0, 1, 0, 0, \ldots, 0] \quad (1)$$

- Sparse vector of size 10,000
- Only one dimension is 1
- No semantic information

Word Embedding

$$\text{“cat”} = [0.27, -0.13, 0.56, \ldots, 0.72] \quad (2)$$

- Dense vector of size 300
- All dimensions have values
- Captures semantic relationships

Embedding Matrix

In neural networks, embeddings are learned via embedding matrices:

$$E \in \mathbb{R}^{V \times d}$$

where V is vocabulary size and d is embedding dimension.

Training Approaches

Supervised Learning

- Use labeled pairs of similar/dissimilar sentences
- Datasets: SNLI, STS-B, Quora Question Pairs
- Minimize similarity loss function

Key Datasets:

- SNLI: Natural Language Inference
- STS-B: Semantic Textual Similarity
- Quora: Question pair similarity

Self-Supervised Learning

- Contrastive learning approach
- No manual labels required
- Create positive pairs via augmentation

Augmentation Techniques:

- Dropout variations
- Token shuffling
- Masking strategies

Training Loss Function

1. Siamese Networks (Contrastive Loss)

$$L = (1 - y) \cdot \max\{0, m - d(x_1, x_2)\}^2 + y \cdot d(x_1, x_2)^2 \tag{3}$$

where $y \in \{0, 1\}$ (1: similar; 0: dissimilar), $d(x_1, x_2) =$ distance, $m =$ margin

2. Triplet Loss (Anchor-Positive-Negative)

$$L = \max\{0, d(x_a, x_p) - d(x_a, x_n) + m\} \tag{4}$$

where x_a, x_p, x_n are embeddings for anchor, positive, and negative sentences

3. Cross-Entropy Loss (Binary Classification)

- Pass sentence pairs through transformer (e.g., BERT)
- Compute similarity score via dot product or cosine
- Predict binary label (similar/not similar)

Modern Embedding Models

Transformer-Based Approaches

1. Initialize with pre-trained models (BERT, T5)
2. Fine-tune on specific similarity tasks
3. Learn to produce similar representations for similar sentences

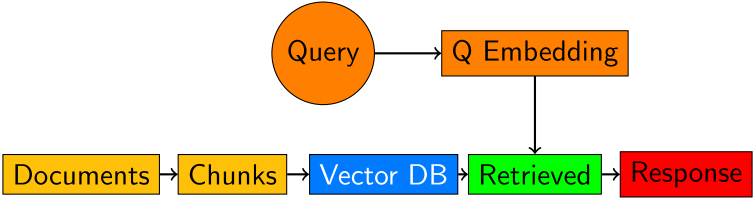
Embedding Models:

- **SBERT**: Sentence-BERT for sentence embeddings
- https://huggingface.co/spaces/mteb/leaderboard

Training Process

The model learns to produce similar vector representations for semantically similar sentences and different representations for dissimilar ones.

RAG System Architecture



Process:

1. Convert text chunks to embeddings: $\text{Chunk}_i \rightarrow \text{Embedding}_{C_i}$
2. Convert question to embedding: $\text{Query} \rightarrow \text{Embedding}_Q$
3. Rank by cosine similarity: $\text{sim}(Q, C_i) = \frac{Q \cdot C_i}{||Q|| \cdot ||C_i||}$
4. Retrieve top-k most similar chunks

Embeddings Enable Information Retrieval

Core Mechanism

Sentence embeddings enable retrieval by ranking chunks based on cosine similarity between question and document embeddings.

Mathematical Process:

$$\text{Query} \rightarrow \text{Embedding}_Q \tag{5}$$

$$\text{Chunk}_1 \rightarrow \text{Embedding}_{C_1} \tag{6}$$

$$\vdots \tag{7}$$

$$\text{Chunk}_n \rightarrow \text{Embedding}_{C_n} \tag{8}$$

Similarity Ranking:

$$\text{Rank} = \text{argsort}[\text{cosine}(Q, C_1), \text{cosine}(Q, C_2), \ldots, \text{cosine}(Q, C_n)]$$

Embeddings represent how the model understands semantic content in text.

Embeddings in Model Validation

1. **Explainability**
   - Embeddings represent model's understanding of semantic features
   - Can be used for interpreting model behavior
   - Visualize decision boundaries and reasoning patterns
2. **Sampling**
   - Diverse and representative sampling via stratified sampling
   - Use embedding clusters for balanced test coverage
   - Ensure comprehensive evaluation across semantic domains
3. **Evaluation**
   - **Context Relevancy**: Match retrieved contexts to prompts
   - **Groundedness**: Match answers to contexts
   - **Answer Relevancy**: Match answers to prompts

Embedding-Based Test Generation

Topic-Driven Testing

Use embedding clusters (topics) to define stratification for comprehensive test coverage.

Test Coverage Strategy:

1. **Random Samples**: Basic sampling from each topic cluster
2. **Twinning Samples**: Guarantee distribution replication

Query Generation Types:

- **Context Handling**: Understanding, recall, reasoning, synthesis
- **Structured Data**: Numbers, tables, charts interpretation
- **Robustness**: Ambiguity, error, noise handling
- **Variability**: Consistent answers across different prompts

Reverse Process

Generate specific query types given context samples to test system capabilities.

Visualizing Embedding: Dimensionality Reduction via UMAP

UMAP (Uniform Manifold Approximation and Projection)

Projects high-dimensional embeddings into low-dimensional space while preserving global and local structure.

Core Principles:

1. **Manifold Learning**: Assumes high-dimensional data lies on lower-dimensional manifold
2. **Graph Theory**: Constructs neighborhood graphs
3. **Optimization**: Minimizes difference between high-dim and low-dim graphs

Key Hyperparameters:

- `n_neighbors`: Local vs global structure focus
- `min_dist`: Cluster density control
- `n_components`: Output dimensionality (2D/3D for visualization)

Clustering and Topic Analysis

Clustering Algorithms:

- **K-means**: Centroid-based clustering
- **HDBSCAN**: Density-based clustering
- **K-medoids**: Robust centroid alternative

Information Extraction:

- **c-TF-IDF**: Cluster-based term importance
- **Representative Samples**: LLM-generated examples

Workflow:

1. Generate embeddings for document chunks
2. Calculate Distance (Apply dimensionality reduction as needed)
3. Perform clustering (K-means/HDBSCAN)
4. Extract keywords using LLM or c-TF-IDF
5. Generate topic summaries with LLM
6. Create stratified samples for evaluation

TF-IDF and c-TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF):

$$TF_{t,d} = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{9}$$

$$IDF_t = \log\left(\frac{N}{1 + DF(t)}\right) \tag{10}$$

$$TF\text{-}IDF_{t,d} = TF_{t,d} \times IDF_t \tag{11}$$

Cluster-based TF-IDF (c-TF-IDF):

$$TF_{t,c} = \frac{\text{Number of term } t \text{ in cluster } c}{\text{Total number of words in cluster } c} \tag{12}$$

$$ICF_t = \log\left(\frac{A}{\text{Number of clusters containing } t}\right) \tag{13}$$

$$c\text{-}TF\text{-}IDF_{t,c} = TF_{t,c} \times ICF_t \tag{14}$$

Embedding Applications

1. **Semantic Search**
   - Find documents based on meaning, not just keywords
   - Cross-lingual information retrieval
   - Question-answering systems
2. **Recommendation Systems**
   - User and item embeddings for collaborative filtering
   - Content-based recommendations
   - Multi-modal recommendations (text + images)
3. **Natural Language Processing**
   - Sentiment analysis
   - Text classification
   - Machine translation
4. **Model Interpretability**
   - Visualizing model decision processes
   - Debugging model behavior

Implementation Considerations

Model Selection:

- Domain-specific vs general-purpose models
- Computational requirements vs accuracy trade-offs
- Language support and multilingual capabilities

Evaluation Metrics:

- Cosine similarity for semantic tasks
- Retrieval metrics: Precision@K, Recall@K, NDCG
- Human evaluation for quality assessment

Scalability Challenges:

- Vector database optimization
- Approximate nearest neighbor search (ANN)
- Batch processing for large-scale embedding generation

Best Practice

Always validate embedding quality on domain-specific tasks before deployment.

Key Takeaways

1. **Embeddings are fundamental** to modern AI systems
    - Convert discrete data to continuous vector representations
    - Capture semantic relationships through spatial proximity
2. **Multiple training approaches** available
    - Supervised learning with labeled data
    - Self-supervised contrastive learning
    - Fine-tuning pre-trained transformer models
3. **Critical for RAG systems**
    - Enable semantic information retrieval
    - Bridge the gap between questions and relevant content
4. **Enable comprehensive evaluation**
    - Topic-based stratified sampling
    - Similarity-based relevance measurement
    - Model interpretability through visualization