Embeddings & Vectors in Machine Learning / AI

---

1. What Are Vectors and Embeddings

Vector

- A *vector* is a list (array) of numbers. In ML/AI, vectors are used to encode features or data points in numeric form.
- Example: 😺 "cat" might be represented as [0.1, 0.7, 0.3, …] in some vector space.

Embedding

- An *embedding* is a special kind of vector representation, often dense (not sparse), learned such that similar things are close together in the vector space. IBM+3GeeksforGeeks+3IBM+3
- Embeddings capture semantics: e.g. similar words, images, or items have embeddings whose vectors are close. Data Forest+1

Why use embeddings?

- They convert high-dimensional or categorical/unstructured data (text, images, graphs) into numerical data ML models can work with. IBM+2GeeksforGeeks+2
- They help reduce dimensions, capture semantic relationships, improve similarity search, clustering, etc. IBM+1

---

2. Types of Embeddings

Type	What Is It	Use-Cases / Examples
Word Embeddings (static)	Each word is mapped to one vector, regardless of context (e.g. Word2Vec, GloVe) Data Forest+2GeeksforGeeks+2	Useful in tasks like sentiment analysis, basic NLP, when context variation is less crucial.
Contextual Word Embeddings	Embeddings depend on sentence/context (so same word gets different vectors depending on where it appears) e.g. BERT, ELMo Salt Data Labs+2GeeksforGeeks+2	Better for tasks needing nuance: word sense disambiguation,

Type	What Is It	Use-Cases / Examples
		translation, question answering.
Sentence / Document Embeddings	Represent whole sentences or documents as one vector. Usually aggregated from word embeddings or via special models (Sentence-BERT etc.) Airbyte+1	Semantic search, document clustering, summarization.
Image Embeddings	Vectors representing images (often from last layers of CNNs or visual models) IBM+1	Image similarity search, classification, content recommendation.
Graph / Node Embeddings	Represent nodes (or edges) in graphs with vectors, preserving graph structure etc. (Node2Vec, Struc2Vec etc.) GeeksforGeeks+2Wikipedia+2	Social network analysis, link prediction, anomaly detection.
Multimodal Embeddings	Embed different data types (text + image, audio + text…) into a shared vector space so comparisons or joint tasks are possible. Airbyte+1	Search across images/text, captioning, recommendation with multiple data sources.

---

3. How Embeddings Are Created / Models & Techniques

- **Word2Vec (Skip-Gram, CBOW)**: Predict words based on context (or vice versa), use large text corpora to learn static word vectors. Data Forest+1
- **GloVe**: Uses global word co-occurrence statistics to learn embeddings. GeeksforGeeks+1
- **FastText**: Extends Word2Vec by including sub-word (character n-gram) information; helps with rare words or morphological variations. GeeksforGeeks
- **Contextual Models (e.g. BERT, ELMo)**: These produce embeddings dependent on the entire input sentence, using transformer or LSTM architectures. Salt Data Labs+2Wikipedia+2

Training / Learning:

- Embedding layers: in neural networks, there's often a layer that maps inputs (word IDs, item IDs, etc.) to embeddings and these embeddings are learned during task-training. Imperial College London+1

- Pre-trained embeddings: use embeddings trained on large corpora, then fine-tune or use as features for downstream tasks. Saves time and improves performance. Imperial College London+1

---

4. Similarity & Operations on Embeddings

- **Similarity metrics**: Cosine similarity, Euclidean distance are used to compare embeddings. If two vectors are close (cosine near 1, or small Euclidean distance), then the original items are semantically similar. Airbyte+1
- **Arithmetic in vector space** (analogy examples): e.g. vector("king") − vector("man") + vector("woman") ≈ vector("queen") in good embedding spaces. Shows that embeddings capture semantic relationships. Imperial College London+1

---

5. Applications of Embeddings & Vectors

Here are real-life uses & examples:

- **Semantic Search / Information Retrieval**: Represent documents and queries as embeddings; rank by similarity. E.g. search engines or QA systems. Airbyte+1
- **Recommendation Systems**: User embeddings + item embeddings to suggest items similar to ones user liked. Softplan | Starian+1
- **Sentiment Analysis / Classification**: Embeddings as input features for classification models. Softplan | Starian
- **Fraud / Anomaly Detection**: Embed transactions or user behavior; detect outliers. Airbyte+1
- **Cross-modal tasks**: e.g. matching image + text; caption generation; image search from text. InterSystems Corporation+1
- **Graph tasks**: Social networks (recommend friends), structural role detection (Struc2Vec), link prediction. Wikipedia+1

---

6. Comparison / Trade-Offs

Feature	Static Embeddings (Word2Vec, GloVe)	Contextual Embeddings (BERT, ELMo etc.)
Same embedding for a word always	✓ Yes	❌ No – depends on context
Simpler, faster	✓ More lightweight	❌ Heavier compute, more complex
Less nuanced	❌ May miss meaning shifts (bank = river bank vs financial bank)	✓ Better at capturing meaning in context
Good for low-resource or simpler tasks	✓	✓ useful but may be overkill sometimes

Also:

- Higher dimensional embeddings capture more nuance but cost more in storage / computation.
- Pre-trained might not cover domain-specific words; fine-tuning or training embeddings for specific domain helps.
- Embeddings may reflect bias in training data; similar words may show gender / racial bias etc.

---

7. Best Practices & Things to Watch Out

- **Dimension choice**: 50-300 dimensions common in word embeddings; higher dims for sentence/image embeddings. Too low - not expressive; too high - inefficient.
- **Normalization**: Sometimes normalize vectors; use cosine similarity as metric rather than raw Euclidean.
- **Handling Out-of-Vocabulary**: For static embeddings, unknown words need fallback strategies (e.g. FastText helps; or embeddings learned on domain).
- **Bias & Fairness**: Check embeddings for biased associations; mitigate if needed.
- **Visualizing embeddings**: Use t-SNE, UMAP etc., to view in 2D/3D to inspect clusters. Helps understand what embedding is doing.

---

8. Summary

- Embeddings are powerful tools: transform complex/unstructured data into numeric vectors that capture meaning, similarity, and context.
- Different kinds (static vs contextual; word/sentence/image/graph) exist depending on what you need.
- Widely used in search, recommendation, classification, etc.
- Need to balance between performance, resource cost, and domain specificity.