Embeddings and Learned Representations

Applied Deep Learning — Chapter 8

Yin Yang

Foundation Books Project

Roadmap

From Raw Objects to Vectors

Representations and Embeddings

Geometry of Embedding Spaces

Image Embeddings from Pretrained CNNs

Token IDs and Word Embeddings

Language Models as Embedding Learners

Bag-of-Embeddings Sentiment

Pretrained Text and Sentence Embeddings

Multimodal Embeddings and CLIP

Evaluating Embeddings

Summary

From Raw Objects to Vectors

Why Embeddings?

Deep learning often becomes useful by mapping complicated raw objects into vectors.

- Images, words, reviews, captions are not vectors at first.
- Once mapped to $\mathbb{R}^d$, ordinary geometry becomes available.
- Compare two vectors, search neighbors, train a small classifier, cache features, inspect errors.

Embedding move: turn raw objects into vectors so ordinary geometry and classifiers can be used.

Two Running Examples

IMDB sentiment

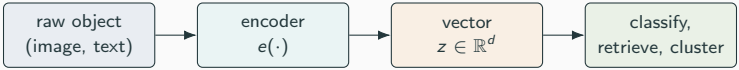
- Review $\rightarrow$ token IDs
- Embedding lookup $\rightarrow$ matrix
- Pooling $\rightarrow$ one review vector
- Dense classifier $\rightarrow$ logit

Image features

- Image $\rightarrow$ pretrained CNN body
- Global average pool $\rightarrow \mathbb{R}^{512}$
- Cache per-image vectors
- Train small head, or retrieve neighbors

Same idea at different scales: raw object $\rightarrow$ vector $\rightarrow$ geometry-based use.

The Pipeline



Examples:

- food image $\rightarrow$ pretrained CNN body $\rightarrow$ retrieve or classify
- movie review $\rightarrow$ token lookup $+$ pooling $\rightarrow$ predict sentiment
- caption $\rightarrow$ text encoder $\rightarrow$ compare with images or documents

Representations and Embeddings

Representations vs. Feature Vectors vs. Embeddings

Representation
A transformed description of an object used by a model. May be a vector, matrix, tensor, or other structured value.

Feature vector
A vector in $\mathbb{R}^d$ whose entries describe an object for a downstream task. Hand-designed or learned.

Embedding function
A map $e : \mathcal{X} \to \mathbb{R}^d$ whose codomain has *useful geometry* — distances, angles, dot products encode task-relevant information.

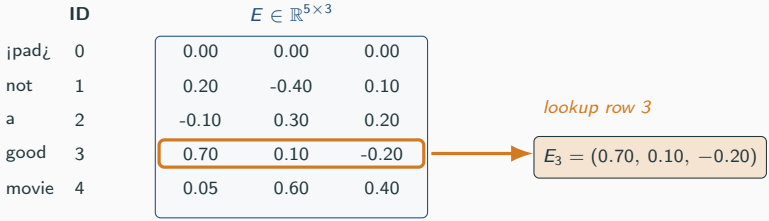
An Embedding Is Not Automatically Good

The definition only says: object $\mapsto$ vector. Quality is empirical.

- Sentiment classifier: does the vector help predict sentiment on held-out reviews?
- Image retrieval: are nearest neighbors useful matches?
- Visualization (PCA, t-SNE, UMAP): a pretty plot is not a substitute for task evaluation.

Vector is not enough. An embedding earns trust only after validation, retrieval, or error checks show useful structure.

Embedding Matrix for Tokens



- Token ID 3 is an *index*, not the number three.
- Real model: $V = 20{,}000$, $d = 128 \Rightarrow Vd = 2{,}560{,}000$ trainable weights.

Geometry of Embedding Spaces

Dot Product, Norm, Distance, Cosine

For $u, v \in \mathbb{R}^d$:

$$u^\top v = \sum_j u_j v_j, \qquad \|u\|_2 = \sqrt{u^\top u}, \qquad \|u - v\|_2.$$

Cosine similarity:

$$\cos(u, v) = \frac{u^\top v}{\|u\|_2 \, \|v\|_2}.$$

- Dot product: large when same direction *and* large magnitude.
- Distance: small when coordinates are close.
- Cosine: angle only — magnitude removed.

Worked Example

Take $u = (1, 2)$, $v = (2, 1)$:

$$u^\top v = 1 \cdot 2 + 2 \cdot 1 = 4, \quad \|u\|_2 = \|v\|_2 = \sqrt{5}.$$

$$\|u - v\|_2 = \sqrt{(1-2)^2 + (2-1)^2} = \sqrt{2}.$$

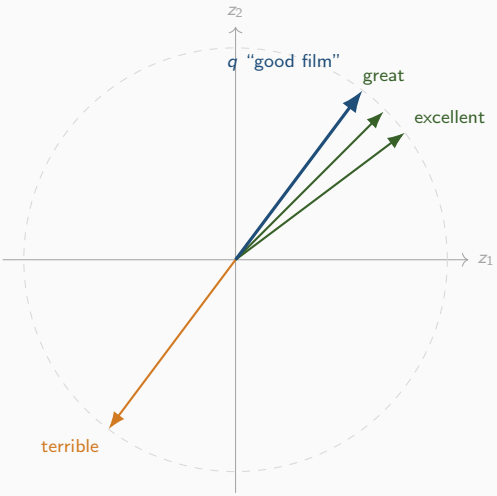
$$\cos(u, v) = \frac{4}{\sqrt{5}\sqrt{5}} = \frac{4}{5} = 0.8.$$

Magnitude effect: $a = (10, 0)$, $b = (1, 0)$, $c = (0, 1)$:

$$a^\top b = 10, \quad b^\top b = 1, \quad \cos(a, b) = 1, \quad \cos(b, c) = 0.$$

Normalize, then dot product = cosine similarity.

Toy Sentiment Retrieval



Query $q = (0.6, 0.8)$:

$$\cos(q, \text{excellent}) = 0.96$$

$$\cos(q, \text{great}) \approx 0.99$$

$$\cos(q, \text{terrible}) = -1.0$$

Nearest-neighbor on the unit circle places positive query near positive phrases; opposite direction is negative.

Coordinates Are Not Interpretable

Real embedding spaces are high-dimensional. Coordinates are usually not individually interpretable.

- No "humor dimension" or "food dimension" without evidence.
- Behavior of the *whole space* under the relevant operation is what matters.
- Retrieval failure: "refund policy" returns ML refunding documents. Don't name a hidden axis. Inspect neighbors, metadata, preprocessing, and similarity rule.

Image Embeddings from Pretrained CNNs

CNN Body as Embedding Function

Let g_θ be a pretrained CNN body and x an image:

$$r = g_\theta(x) \in \mathbb{R}^d.$$

- Feature map $7 \times 7 \times 512 \to$ global average pool $\to \mathbb{R}^{512}$.
- $n = 8{,}000$ images $\to$ feature table $8{,}000 \times 512$.
- Train a small classifier on the table: fast, easy ablation.
- Use vectors for nearest-neighbor retrieval as well.

Layer choice matters. Body just before the classifier head is a strong first try; very early layers $\approx$ raw patterns; final logits are too task-specialized.

Caching Discipline

Caching changes the workflow.

- Decide split before any tuning.
- Same frozen body for train and validation feature extraction.
- Hyperparameters chosen by validation only.
- Test set untouched until final reporting.

Retrieval as a debug tool: if a soup query returns cake images, the embedding may be poor for the domain — or preprocessing is wrong.

Token IDs and Word Embeddings

Tokens, Vocabulary, IDs

Definitions
A *token* is a discrete text unit. A *vocabulary* is a finite set of tokens. A *token ID* is an integer index.

Tokenization is configuration, not formatting:

- Lowercasing merges `Great` and `great`.
- Keeping punctuation may let `great!` signal stronger sentiment.
- Subword pieces reduce out-of-vocabulary issues but lengthen sequences.

Token ID is an index, not a measurement. `good`=77 and `bad`=142 does not mean "*bad* > *good*."

Embedded Review Shape

Review "not a good movie" $\rightarrow$ IDs $(95, 7, 77, 19)$. Embedded review (with $E \in \mathbb{R}^{V \times d}$, $d = 64$):

$$\begin{bmatrix} E_{95} \\ E_{7} \\ E_{77} \\ E_{19} \end{bmatrix} \in \mathbb{R}^{4 \times 64}.$$

Batch of 32 reviews padded/truncated to length 200:

$$32 \times 200 \times 64.$$

Sparse row updates: a mini-batch produces gradient only for the rows of token IDs it actually contains.

Word Vectors Learned by Prediction

word2vec, GloVe, fastText share a lesson: useful word geometry is learned from patterns in unlabeled text.

- word2vec CBOW: predict target word from context.
- word2vec skip-gram: predict context from target.
- GloVe: global co-occurrence statistics.
- fastText: subword pieces help rare/morphological words.

Limitation: a single word vector does not solve sentence meaning. “not good” depends on the relationship between two tokens.

Language Models as Embedding Learners

Prediction Teaches Representation

Language model
A model that assigns probabilities to text sequences. Training often means predicting a missing or next token from context.

Next-token loss over $(w_1, \ldots, w_T)$:

$$\mathcal{L}_{\text{next}} = -\sum_{t=1}^{T-1} \log p_\theta(w_{t+1} \mid w_1, \ldots, w_t).$$

Masked-token loss over masked positions M:

$$\mathcal{L}_{\text{mask}} = -\sum_{t \in M} \log p_\theta(w_t \mid w_{\text{visible}}).$$

Self-supervised: target is created from the text itself.

From Word Vectors to Large Language Models

- Same broad idea, different scale.
- word2vec/GloVe: small word table for narrow tasks.
- Modern LLMs: very large neural networks on massive text corpora, typically next-token prediction.
- Result: not just token embeddings but a stack of useful internal representations.
- Transformer architecture made this scaling especially effective.

This chapter does not yet need the architecture — only the bridge: language-model training is a powerful way to turn raw text into reusable vectors.

Bag-of-Embeddings Sentiment

Bag of Embeddings: The Baseline

Token embeddings $e_1, \ldots, e_T \in \mathbb{R}^d$. Average pooling:

$$\bar{e} = \frac{1}{T} \sum_{t=1}^{T} e_t.$$

Worked example: "good movie" with $e_{\texttt{good}} = (0.8, 0.2)$, $e_{\texttt{movie}} = (0.1, 0.4)$:

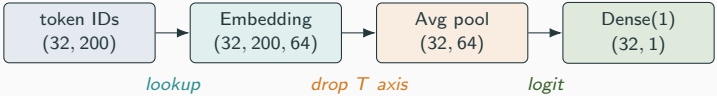
$$\bar{e} = (0.45, 0.30).$$

Classifier with $w = (2, -1)$, $b = -0.2$:

$$s = w^\top \bar{e} + b = 2(0.45) - 1(0.30) - 0.2 = 0.4.$$

Positive logit $\Rightarrow$ probability > 0.5 after sigmoid.

Shape Trace as Debugging



- Trace shapes first. Most embedding bugs show up as wrong batch, length, or dimension.
- Padding is not text. Pool only over real tokens; otherwise short reviews drift toward the padding vector.

IMDB Reference Run (validation)

Stratified 80/20 split of ACL IMDB train, seed 1234. 20,000 train, 5,000 validation.

Model	Params	Time	Acc.	Loss
Majority class	0	–	0.5000	–
TF-IDF linear	20,001	12.3s	0.8890	0.3107
Bag-of-emb $d = 64$	1,280,065	10.8s	0.8824	0.2873
Bag-of-emb $d = 128$	2,560,129	14.1s	0.8902	0.2694
Bag-of-emb len 200	1,280,065	9.8s	0.8650	0.3113
Bag-of-emb hidden 64	1,284,225	8.4s	0.8936	0.2705

Numbers are evidence for this split and hardware — not universal constants.

Order Is Gone

- Average pooling discards token order.
- “good, not boring” and “boring, not good” have similar tokens but different meaning.
- Predictable failure pattern: negation, contrast, composition.

This motivates sequence models (RNNs, Transformers): keep the embedded review matrix and read it as an ordered sequence.

Pretrained Text and Sentence Embeddings

Pretrained Embeddings: Three Reuse Patterns

Pretrained embedding
An embedding function or table whose parameters were learned before the current target-task training run.

Three uses for a pretrained word table:

1. Freeze — vectors do not change during sentiment training.
2. Fine-tune — adapt vectors to the task; risk of overfitting.
3. Initialize, then train with a smaller learning rate.

Vocabulary coverage matters — record it.

Sentence and Document Embeddings

Modern systems often map an entire sentence or document to one vector:

$$z_{\text{text}} = e(\text{text}) \in \mathbb{R}^d.$$

Examples: Sentence-BERT, Universal Sentence Encoder.

- Logistic regression on review vectors.
- Semantic search by cosine similarity.
- Clustering reviews to inspect themes.

Caching is responsibility: stale cache when encoder or preprocessing changes; validation must remain independent of model selection.

Multimodal Embeddings and CLIP

Shared Embedding Space

Definition
A shared embedding space maps objects from ≥ 2 modalities so that cross-modal comparisons are meaningful.

Image encoder f_{image}, text encoder f_{text}:

$$z_{\text{image}} = \frac{f_{\text{image}}(x)}{\|f_{\text{image}}(x)\|_2}, \qquad z_{\text{text}} = \frac{f_{\text{text}}(t)}{\|f_{\text{text}}(t)\|_2}.$$

Score:

$$z_{\text{image}}^{\top} z_{\text{text}}.$$

Both vectors normalized $\Rightarrow$ dot product $=$ cosine.

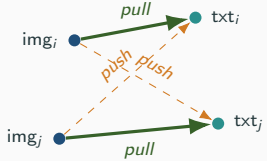
CLIP Toy Numbers

Image vector $z_{\text{image}} = (0.6, 0.8)$:

text prompt	z_1	z_2	similarity
`a bowl of soup`	0.8	0.6	0.96
`a chocolate cake`	-0.2	0.98	0.66
`a sports car`	-0.8	-0.6	-0.96

Image is most similar to the soup prompt. Real CLIP uses much higher dimensions and learned encoders, but the scoring operation is the same.

CLIP Contrastive Loss



Score $s_{ij} = z_i^{\text{img}\top} z_j^{\text{txt}} / \tau$.

$$\mathcal{L}_{\text{i2t}} = -\frac{1}{B} \sum_i \log \frac{\exp(s_{ii})}{\sum_j \exp(s_{ij})}.$$

Diagonal s_{ii}: matched pair (pull). Off-diagonal: contrastive alternative (push).

Uses: zero-shot classification, text-to-image search, image-to-text search.

CLIP Cautions

Powerful tools, but still ranking results.

- Prompt wording changes outcomes.
- Training data may differ from target domain.
- Similarity can reflect superficial correlations.
- Biases in training data appear in retrieval.

Need: task-specific evaluation, subgroup checks when people are affected, error inspection.

Evaluating Embeddings

Evaluate by the Task

No universal embedding score.

Classification validation accuracy, loss, parameter count, time.

Retrieval precision@k, recall@k, top-k accuracy, MRR.

Inspection duplicates, mislabeled examples, meaningful clusters.

Toy example with 4 relevant items, top-5 pattern R, N, R, R, N:

$$P@5 = 3/5 = 0.60, \quad R@5 = 3/4 = 0.75, \quad \text{MRR} = 1.$$

Leakage Is the Common Failure

Data leakage
Information that should be unavailable during training or model selection influences the fitted model, preprocessing, or hyperparameters.

Examples in this chapter:

- Validation labels inform preprocessing decisions.
- Kaggle public leaderboard checked repeatedly while tuning.
- Cached embeddings mixed across split boundaries.

Leakage flatters models. Keep validation and test info out of preprocessing, tuning, and cache choices.

Reading Validation Mistakes

Error analysis is the main qualitative check.

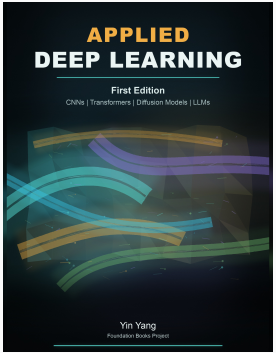
Metric shift	First explanation to test
Long-review errors increase	decisive sentence past length limit
Vocab expansion helps	rare words mapped to UNK
Train up, val down	overfitting frequent phrases
Negation still wrong	order-sensitive meaning lost in pooling
Leaderboard improves with submissions	leaderboard became selection signal

Summary

Summary

- **Embeddings** = learned vector representations.
- **CNN body** (image $\rightarrow$ feature vector) and **embedding matrix** (token ID $\rightarrow$ word vector) are the same idea.
- **Geometry:** dot product, distance, cosine similarity. Normalized vectors $\Rightarrow$ cosine = dot product.
- **Bag-of-embeddings** is fast and clear — but loses order.
- **Pretrained / sentence / CLIP** extend the same operation to search, classification, clustering, cross-modal similarity.
- **Validation rules:** task-specific metrics, leakage control, bias checks, error analysis.

Next: keep the embedded review matrix, but read it as a sequence — recurrent networks and attention.



Applied Deep Learning

Chapter overview slides for the book by Yin Yang.

Book page	`foundation-books.github.io/applied-deep-learning`
Code	`github.com/foundation-books/applied-deep-learning-code`
Print	`amazon.com/dp/B0GZF7QRSH`
Kindle	`amazon.com/dp/B0GZF9221X`
License	CC BY-NC-ND 4.0

These free overview slides may be shared non-commercially with attribution and without modifications. Full explanations, exercises, and companion-code context are in the book and linked repository.