Vector Semantics & Embeddings — Condensed Notes

Koorosh Asil Ghrehbaghi
K.N. Toosi University of Technology — B.Sc. Computer Science
Instructor: Dr. Maryam Abdolali

September 13, 2025

Abstract

Condensed and complete chapter notes covering *Vector Semantics and Embeddings*: lexical semantics, co-occurrence matrices (term-document, term-term), similarity (dot/cosine), weightings (tf-idf, PPMI), dense embeddings (word2vec SGNS), optimization and training, visualization, and practical considerations including bias. Includes formulas, derivations, pseudocode, and a final checklist of every concept you should learn from the chapter.

Contents

1 Lexical Semantics — core ideas

Definition

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings.

Definition

Lemma / Wordform / Sense: A *lemma* (citation form) groups inflected wordforms; a *sense* is a distinct meaning of a lemma (polysemy).

Note

Lexical relations: synonymy, antonymy, hypernymy (IS-A), meronymy (part-whole), relatedness (associations), semantic fields and frames (roles like buyer/seller).

2 Vector semantics — main idea

Definition

Embedding / vector semantics: Represent each word as a point (vector) in a high-dimensional space derived from distributional statistics. Similar meanings → nearby vectors.

Important

Embeddings are the representation-learning backbone of modern NLP (static embeddings like word2vec/GloVe/fastText and contextual embeddings like BERT).

3 Co-occurrence matrices

3.1 Term–document matrix

Rows: terms; Columns: documents. Each cell: count(term, document). Useful for IR (tf-idf weighting).

3.2 Term–term (word–context) matrix

Rows: target words; Columns: context words (windowed). Each cell: number of times context word occurs near target.

Example

If the window is ± 4 words, cell (w, c) counts occurrences of c within 4 tokens of w across corpus.

4 Similarity: dot product and cosine

Definition

Dot product / inner product: For vectors $v, w \in \mathbb{R}^n$,

$$v \cdot w = \sum_{i=1}^{n} v_i w_i.$$

Definition

Vector length / norm: $\|v\| = \sqrt{\sum_i v_i^2}$.

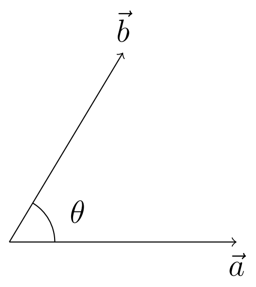
Definition

Cosine similarity:

$$\cos\theta(v, w) = \frac{v \cdot w}{\|v\|\|w\|}.$$

Note

Cosine normalizes for length (frequency). For nonnegative-count vectors, $\cos \in [0, 1]$ typically.



5 Weightings: TF–IDF

Definition

Term frequency (tf): raw $tf_{t,d} = \text{count}(t, d)$ or log-weighted:

$$\text{tf}_{t,d} = \log(\text{count}(t, d) + 1).$$

Definition

Inverse document frequency (idf):

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t},$$

where N is total number of documents and df_t is number of documents containing t.

Definition

TF–IDF weight (product):

$$w_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t.$$

Note

TF–IDF reduces weight of common (non-discriminative) terms like "the" and raises weight of document-specific words like "Romeo".

6 Pointwise Mutual Information (PMI) and PPMI

Definition

Pointwise Mutual Information (PMI): For target word w and context c,

$$\mathrm{PMI}(w, c) = \log \frac{P(w, c)}{P(w)P(c)}.$$

Compute probabilities via MLE from co-occurrence counts.

Definition

Positive PMI (PPMI):

$$\mathrm{PPMI}(w, c) = \max(\mathrm{PMI}(w, c), 0).$$

Note

PMI highlights surprising co-occurrences; PPMI removes unreliable negative PMI (rare-event noise).

Important

Variants: smoothing contexts via $P_\alpha(c) \propto \mathrm{count}(c)^\alpha$ (e.g., $\alpha = 0.75$) helps with rare contexts.

7 From sparse to dense embeddings

Note

Sparse vectors: long ($|V|$ dims), mostly zeros (tf-idf, PPMI). Dense embeddings: short (50–1000 dims), real-valued, learned (word2vec, GloVe, fastText). Dense usually outperforms sparse in downstream tasks.

8 Word2Vec — Skip-gram with Negative Sampling (SGNS)

8.1 Intuition

Train a classifier that, given a (target, context) pair (w, c), predicts whether c is observed in the context of w. Learn embeddings as classifier parameters.

8.2 Model: similarity & probability

Let v_w be target (input) embedding for w and u_c be context (output) embedding for c. Use dot product as similarity and sigmoid to convert to probability:

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

Model:

$$P(+ \mid w, c) = \sigma(u_c^\top v_w).$$

8.3 Negative sampling objective (per positive pair)

For a positive pair (w, c^+) and k negative samples $\{c_1^-, \dots, c_k^-\}$ drawn from noise distribution $P_n(c)$ (commonly proportional to count$^{0.75}$), the per-instance loss to *minimize* is:

$$\mathcal{L} = -\log \sigma(u_{c^+}^\top v_w) - \sum_{i=1}^{k} \log \sigma(-u_{c_i^-}^\top v_w).$$

Equivalent: maximize $\log \sigma(u_{c^+}^\top v_w) + \sum \log \sigma(-u_{c^-}^\top v_w)$.

8.4 Gradients (compact forms)

For positive pair (w, c^+):

$$\frac{\partial \mathcal{L}}{\partial v_w} = (\sigma(u_{c^+}^\top v_w) - 1)\, u_{c^+} \; + \; \sum_{i=1}^{k} \sigma(u_{c_i^-}^\top v_w)\, u_{c_i^-}.$$

For a single positive and its negative samples, the per-sample update for stochastic gradient descent uses these gradients; similarly compute gradients for u_c vectors.

8.5 Noise distribution

$$P_n(w) \propto \text{count}(w)^{0.75}.$$

This downweights extremely common words relative to raw unigram sampling.

8.6 SGNS pseudocode

Listing 1: Skip-gram with Negative Sampling (outline)

```
Initialize v_w, u_w ~ small random values for every word w in V
for epoch = 1..E:
  for each position t in corpus with target word w = word[t]:
    contexts = window around t
    for each context c in contexts:
      sample k negative words {c_neg} ~ P_n(.)
      # Compute gradients and apply updates (SGD / Adam)
      # Update v_w, u_c, and each u_c_neg using derivatives of:
      #  - log sigma(u_c^T v_w) - sum log sigma(-u_c_neg^T v_w)
```

Note

Implementation notes: minibatching, subsampling of frequent words, vector addition of input+output embeddings (or use only input embeddings) are common heuristics. Pretrained code and models available (word2vec, GloVe, fastText).

9 Other static embeddings

Definition

GloVe: Global Vectors — factorizes a (weighted) log co-occurrence matrix; learns dense vectors by optimizing an objective motivated by ratios of co-occurrence probabilities.

Definition

fastText: constructs word vectors from character n-gram embeddings (handles morphologically-rich languages and OOV words).

10 Applications and evaluation

- Word similarity and relatedness (cosine; datasets like SimLex-999).
- Analogy tasks: solve $b^* \approx \arg\max_x \cos(v_x, v_b - v_a + v_{a^*})$ (e.g., king - man + woman $\approx$ queen).

- Downstream: classification, retrieval, QA, summarization, clustering.
- Document vectors: centroid of word vectors or weighted centroid (tf-idf).

11 Visualization and properties

- Dimensionality reduction (PCA, t-SNE) to 2D for plotting.
- Clustering or nearest neighbors listing for inspecting semantics.
- Observed properties: linear structure (analogies), sensitivity to context window (smaller window = syntactic similarity, larger window = topical similarity).

12 Bias, stability, and best practices

Note

Embeddings reflect and can amplify social biases present in training data (gender, race, age). Evaluate and mitigate (debiasing transforms, curated datasets, monitoring). Train multiple random restarts and average or bootstrap when analyzing corpus-specific associations.

13 Concise derivations (key math)

13.1 Cosine normalization

From dot-product similarity $v \cdot w$, divide by norms:

$$\frac{v \cdot w}{\|v\|\|w\|} = \cos\theta.$$

13.2 SGNS objective intuition

Maximize probability of observed (target,context) pairs and minimize probability of randomly sampled noise pairs. Negative sampling approximates a full softmax-based objective efficiently.

14 Complete checklist — *learn everything* from this chapter

1. **Key definitions:** distributional hypothesis, lemma, wordform, word sense, semantic field, semantic frame, embedding.
2. **Co-occurrence representations:** term–document matrix and term–term (word–context) matrix.

3. **Linear algebra basics:** vector, dot product, norm, cosine similarity — formulas and intuition.
4. **Weightings:** raw counts, log tf, idf formula and intuition, tf–idf product and role in IR.
5. **Mutual information:** PMI formula, meaning of PMI, PPMI and why negative values are clipped.
6. **Probability estimation:** how to convert co-occurrence counts to probabilities and joint marginals.
7. **PPMI smoothing variants:** context power α (e.g., 0.75), Laplace smoothing.
8. **Similarity computations:** how to compute nearest neighbors and document centroids (Eq. for centroid).
9. **Dense embeddings:** what they are, dimensionality d, benefits vs sparse vectors.
10. **Word2Vec (skip-gram):**
    - model form $P(+|w,c) = \sigma(u_c^\top v_w)$,
    - negative sampling objective: $\mathcal{L} = -\log \sigma(u_{c^+}^\top v_w) - \sum \log \sigma(-u_{c^-}^\top v_w)$,
    - noise distribution $P_n(w) \propto \text{count}(w)^{0.75}$,
    - gradient forms and SGD updates,
    - practical hyperparameters: window size L, negative samples k, embedding size d, learning rate, subsampling.
11. **Alternatives / variants:** CBOW, GloVe, fastText — their core ideas.
12. **Evaluation tasks:** word similarity, analogy tasks, downstream transfer tasks (classification, retrieval).
13. **Visualization:** PCA / t-SNE, clustering and nearest-neighbor inspection.
14. **Bias and ethics:** how embeddings encode social bias, methods to detect and mitigate (debiasing).
15. **Practical tips:** pre-trained models, bootstrapping / multiple runs to check stability, use tf-idf weighting for document centroids.
16. **Extra:** connection between count-based methods and predictive models (word2vec approximates shifted PPMI).

15 Short list of formulas (one-line reference)

- Dot product: $v \cdot w = \sum_i v_i w_i$.
- Norm: $\|v\| = \sqrt{\sum_i v_i^2}$.
- Cosine: $\cos(v, w) = \dfrac{v \cdot w}{\|v\|\|w\|}$.
- TF (log): $\mathrm{tf}_{t,d} = \log(\mathrm{count}(t, d) + 1)$.
- IDF: $\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$.
- TF–IDF: $w_{t,d} = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_t$.
- PMI: $\mathrm{PMI}(w, c) = \log \frac{P(w,c)}{P(w)P(c)}$.
- PPMI: $\mathrm{PPMI} = \max(\mathrm{PMI}, 0)$.
- Sigmoid: $\sigma(x) = \frac{1}{1+\exp(-x)}$.
- SGNS loss (per positive pair): $\mathcal{L} = -\log \sigma(u_{c^+}^\top v_w) - \sum_{i=1}^{k} \log \sigma(-u_{c_i^-}^\top v_w)$.

16 Further reading

- Pennington, Socher & Manning (GloVe), Mikolov et al. (word2vec), Bojanowski et al. (fastText).
- Jurafsky & Martin — Chapter on Vector Semantics (this chapter): for more depth and exercises.
- Levy & Goldberg (connections between word2vec and PPMI).

Note

If you want this shorter or longer, or prefer a slide-style (beamer) version, or a printable one-page cheat-sheet extracted from the checklist above, tell me the format and I'll produce it immediately.