

Word vectors

Jan 20, 2026

Outline

1 Word meaning

2 Word vector

3 Word2vec

4 Comments

5 Appendix

Outline

1 Word meaning

2 Word vector

3 Word2vec

4 Comments

5 Appendix

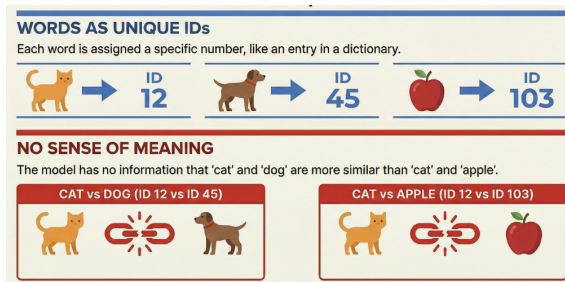
Last class: What is NLP?

Natural Language Processing (NLP) is a subfield of computer science and artificial intelligence that develops computational methods for enabling computers to understand, interpret, and generate human language.

- What does it mean to understand human language?
- Q. How do we represent **meanings** in computer?

Encoding: Background

- In traditional NLP, people treated words as discrete symbols.



- Practically building/updating a database is expensive.

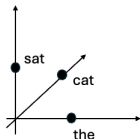
Encoding

- Words themselves cannot be given as inputs to computers
- **Numbers** can be.
- **Encoding**: converting words to numbers
- **Vector**: an ordered list of numbers (e.g., [0.1, 0.3, -0.5])

One-hot encoding: Problem

■ *the cat sat*

word	encoding
the	[1, 0, 0]
cat	[0, 1, 0]
sat	[0, 0, 1]



- Vectors must be as large as the vocabulary size (high dimensional).
- Still, words carry no information about meaning or similarity.
- All one-hot vectors are orthogonal (equally distant from each other)
- **Cosine similarity:**

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \|\vec{B}\|}$$

One-hot encoding: Solution

- Move from **sparse** to **distributed** representation
- Learn to encode similarity in the vectors themselves
- Word **embeddings**

Two viewpoints on meaning

There are different theoretical views on what it means to “know” the meaning of a word (Manning, 2022)

- 1 Denotational semantics** Meaning is the set of objects, events, or situations in the world that a word or sentence refers to.
- 2 Distributional semantics** Meaning is characterized by the **contexts** in which a word appears.

Meaning and language models

- Hypothesis: If meaning is defined as patterns of use and relations among words, then pretrained language models *do* learn meanings.

...government debt problems turning into **banking** crises as happened in 2009...

...saying that Europe needs unified **banking** regulation to replace the hodgepodge...

...India has just given its **banking** system a shot in the arm...

These **context words** will represent **banking**

- “You shall know a word by the company it keeps” (Firth, 1957): One of the most successful ideas of modern statistical NLP.

Sourced from: Manning, C. D. (2022). Human language understanding & reasoning. *Daedalus*, 151(2), 127-138.

Outline

1 Word meaning

2 Word vector

3 Word2vec

4 Comments

5 Appendix

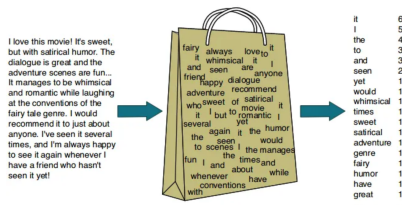
Two different approaches

- 1 Count-based models
- 2 Neural network-based models

Count-based Models

1. Start with a Bag-of-Words (BoW) representation

Bag of Words



(source: <https://nachi-keta.medium.com/nlp-explain-bag-of-words-3b9fc4f211e8>)

- Bag-of-Words assumption: Context words are treated as unordered.
- In other words, the position of a context word relative to the target is ignored.

Count-based Models

2. Extend to a **co-occurrence matrix**: count how often words appear together in a context window

Example sentences:

- I like apples.
- You like bananas.
- They eat bananas.
- We enjoy apples.
- They like fruit.

Example sentences:

- I like apples.
- You like bananas.
- They eat bananas.
- We enjoy apples.
- They like fruit.

	I	you	we	they	like	eat	enjoy	apples	bananas	fruit
I	0	0	0	0	1	0	0	0	0	0
you	0	0	0	0	1	0	0	0	0	0
we	0	0	0	0	0	0	1	0	0	0
they	0	0	0	0	1	1	0	0	0	0
like	1	1	0	1	0	0	0	1	1	1
eat	0	0	0	1	0	0	0	0	1	0
enjoy	0	0	1	0	0	0	0	1	0	0
apples	0	0	0	0	1	0	1	0	0	0
bananas	0	0	0	0	1	1	0	0	0	0
fruit	0	0	0	0	1	0	0	0	0	0

Co-occurrence Matrix (window size = 1)

Count-based Models

3. Apply Singular Value Decomposition (SVD) to reduce dimensions (i.e., a way of breaking a big matrix into a smaller pieces)

Count-based models (Limitations)

■ High computational cost

- Co-occurrence matrix size: $|V| \times |V|$ (vocabulary squared)
- Too large to store/compute for big corpora

■ Sparse and noisy

- Most cells are 0 \Rightarrow sparse matrix
- Rare words/contexts yield unreliable statistics

■ Poor scalability / update issues

- Adding new words requires recomputing the entire matrix and SVD

Neural Network–Based Models

- Count-based models
- **Neural models:** learn vectors *directly* by predicting context words + neural network

Neural Network–Based Models

- Count-based models
- **Neural models:** learn vectors *directly* by predicting context words + neural network

Consistent progress

- 1986: *Learning representations by back propagating errors* (Rumelhart et al., 1986)
- 2003: *A neural probabilistic language model* (Bengio et al., 2003)
- 2013: **Word2Vec** (Skip-gram, CBOW)
- 2014–2015: **GloVe**, fastText
- 2018– : Contextual embeddings (ELMo, BERT, GPT)

Today's presentations

- Emily: Mikolov et al. (2013). *Efficient Estimation of Word Representations in Vector Space*. (Word2Vec)
- Sindhu: Pennington et al. (2014). *GloVe: Global Vectors for Word Representation* (GloVe)

Outline

1 Word meaning

2 Word vector

3 Word2vec

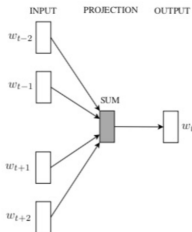
4 Comments

5 Appendix

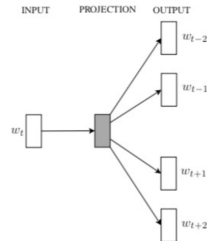
Word2vec: Overview

- **Word2vec** (Mikolov et al., 2013) is a framework for learning word vectors
- Idea:
 - Start with a large corpus (“body”) of text
 - Every word in a fixed vocabulary is represented by a **vector**
 - Go through each position t in the text, which has a center word c and context word o (window, n-gram)
 - Use the similarity of the word vectors for c and o to **calculate the probability** of o given c (or vice versa) ← **Neural network-based model**

Word2vec: Two models



Continuous Bag of Words (CBOW):
predicting the center words using
the context words ($P(w_t | w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$)



Skip-grams (SG):
predicting the context words using
the center word ($P(w_{t+i} | w_t), i \in \{-2, -1, 1, 2\}$)

Focus on **Skip-gram**.

Word2Vec: Skip-grams (*window size = 1*)

- “king brave man”
- “queen beautiful woman”

word	neighbor
king	brave
brave	king
brave	man
man	brave
queen	beautiful
beautiful	queen
beautiful	woman
woman	beautiful

Word2Vec: Skip-grams (*window size = 2*)

- “king brave man”
- “queen beautiful woman”

word	neighbor
king	brave
king	man
brave	man
brave	king
man	king
man	brave
queen	beautiful
queen	woman
beautiful	queen
beautiful	woman
woman	queen
woman	beautiful

Word2Vec: Skip-grams (window size = 2)

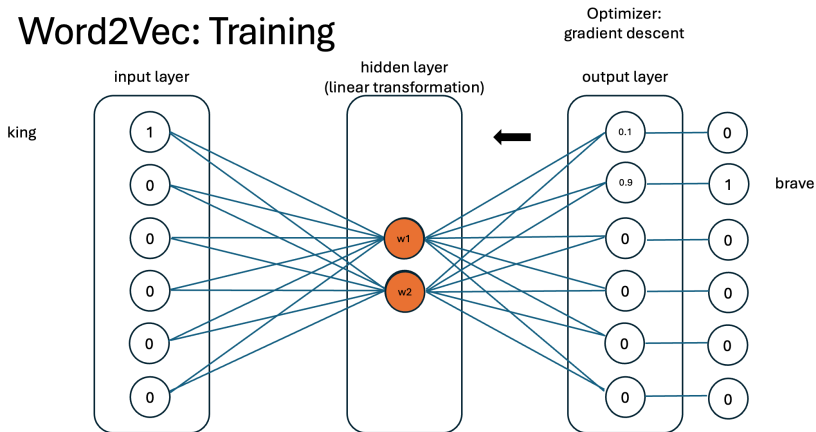
word	one-hot encoding	neighbor	one-hot encoding
king	[1, 0, 0, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
king	[1, 0, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	man	[0, 0, 1, 0, 0, 0]
brave	[0, 1, 0, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	king	[1, 0, 0, 0, 0, 0]
man	[0, 0, 1, 0, 0, 0]	brave	[0, 1, 0, 0, 0, 0]
queen	[0, 0, 0, 1, 0, 0]	beautiful	[0, 0, 0, 0, 1, 0]
queen	[0, 0, 0, 1, 0, 0]	woman	[0, 0, 0, 0, 0, 1]
beautiful	[0, 0, 0, 0, 1, 0]	queen	[0, 0, 0, 1, 0, 0]
beautiful	[0, 0, 0, 0, 1, 0]	woman	[0, 0, 0, 0, 0, 1]
woman	[0, 0, 0, 0, 0, 1]	queen	[0, 0, 0, 1, 0, 0]
woman	[0, 0, 0, 0, 0, 1]	beautiful	[0, 0, 0, 0, 1, 0]

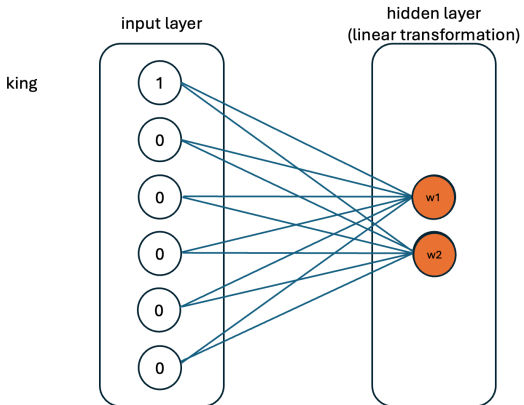
Word2Vec: Input and output

input
[1, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1]

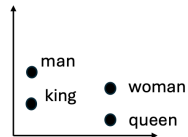
output
[0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0]
[0, 0, 1, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 1, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 1]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 0]

Word2Vec: Training





word	embedding
king	[1, 1]
brave	[1, 2]
man	[1, 3]
queen	[5, 1]
beautiful	[5, 2]
woman	[5, 3]



1. One-hot and embedding lookup

- Each word in the vocabulary is represented as a **dense vector**.
- All these word vectors are stored in a single matrix:

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

2. Predicting context words

- Take the center word's embedding
- Compare it with each candidate context word's output vector
- Compute a **dot product** as a similarity score (see Appendix)

3. From similarity scores to probabilities

- After retrieving the center word and a context word's vectors, we compute their **dot product**:

$$\text{score} = \vec{v}_c \cdot \vec{u}_w$$

3. From similarity scores to probabilities

- After retrieving the center word and a context word's vectors, we compute their **dot product**:

$$\text{score} = \vec{v}_c \cdot \vec{u}_w$$

- To interpret this score as a probability, we apply the **sigmoid function**:

$$\sigma(\text{score}) = \frac{1}{1 + e^{-\text{score}}}$$

3. From similarity scores to probabilities

- After retrieving the center word and a context word's vectors, we compute their **dot product**:

$$\text{score} = \vec{v}_c \cdot \vec{u}_w$$

- To interpret this score as a probability, we apply the **sigmoid function**:

$$\sigma(\text{score}) = \frac{1}{1 + e^{-\text{score}}}$$

- The output is a number between 0 and 1 — representing how likely this word is to appear in the context.

4. Compute loss

- We compare predicted probabilities with actual labels:

4. Compute loss

- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1

4. Compute loss

- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1
 - **Negative (random) words** \rightarrow label = 0

4. Compute loss

- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1
 - **Negative (random) words** \rightarrow label = 0
- We apply the **binary cross-entropy loss**:

$$\mathcal{L} = - \left(\log \sigma(\vec{v}_c \cdot \vec{u}_{w^+}) + \sum_{i=1}^k \log \left(1 - \sigma(\vec{v}_c \cdot \vec{u}_{w_i^-}) \right) \right)$$

4. Compute loss

- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1
 - **Negative (random) words** \rightarrow label = 0
- We apply the **binary cross-entropy loss**:

$$\mathcal{L} = - \left(\log \sigma(\vec{v}_c \cdot \vec{u}_{w^+}) + \sum_{i=1}^k \log \left(1 - \sigma(\vec{v}_c \cdot \vec{u}_{w_i^-}) \right) \right)$$

- The model is rewarded when:

4. Compute loss

- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1
 - **Negative (random) words** \rightarrow label = 0
- We apply the **binary cross-entropy loss**:

$$\mathcal{L} = - \left(\log \sigma(\vec{v}_c \cdot \vec{u}_{w^+}) + \sum_{i=1}^k \log \left(1 - \sigma(\vec{v}_c \cdot \vec{u}_{w_i^-}) \right) \right)$$

- The model is rewarded when:
 - It assigns high probability to true context words

4. Compute loss

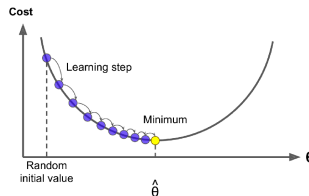
- We compare predicted probabilities with actual labels:
 - **True context words** \rightarrow label = 1
 - **Negative (random) words** \rightarrow label = 0
- We apply the **binary cross-entropy loss**:

$$\mathcal{L} = - \left(\log \sigma(\vec{v}_c \cdot \vec{u}_{w^+}) + \sum_{i=1}^k \log \left(1 - \sigma(\vec{v}_c \cdot \vec{u}_{w_i^-}) \right) \right)$$

- The model is rewarded when:
 - It assigns high probability to true context words
 - It assigns low probability to negative (random) words

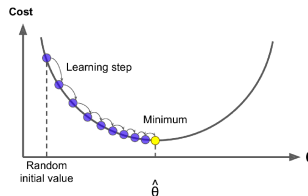
5. Update word vectors

- Optimizer updates parameters based on gradients



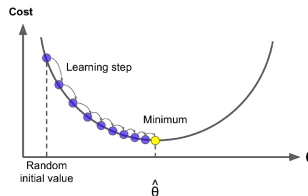
5. Update word vectors

- Optimizer updates parameters based on gradients
- Parameters updated:



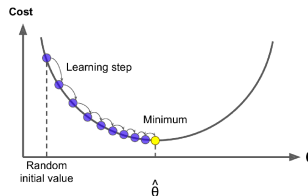
5. Update word vectors

- Optimizer updates parameters based on gradients
- Parameters updated:
 - The center word's vector



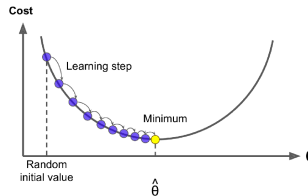
5. Update word vectors

- Optimizer updates parameters based on gradients
- Parameters updated:
 - The center word's vector
 - The true context word's vector
 - The negative samples' vectors



5. Update word vectors

- Optimizer updates parameters based on gradients
- Parameters updated:
 - The center word's vector
 - The true context word's vector
 - The negative samples' vectors
- Over time, words with similar contexts move closer



We'll discuss more about the neural network next week.

Outline

1 Word meaning

2 Word vector

3 Word2vec

4 Comments

5 Appendix

From mini survey

- Q. Where to go to request an official doc for late assignment?
- A: Please email me first.

Thursday

- **Thursday:** Lab 2 — Word2Vec and GloVe; Presenter: Leona

Paper presentation

Please upload your slides on mycourses → assignment → presentation - BEFORE your presentation.

Outline

1 Word meaning

2 Word vector

3 Word2vec

4 Comments

5 Appendix

Dot product as similarity score

- **Algebraic definition:** For two vectors $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$,

$$a \cdot b = \sum_{i=1}^n a_i b_i$$

(multiply each coordinate and add them up)

- **Geometric interpretation:** The same dot product can also be written as

$$a \cdot b = \|a\| \|b\| \cos \theta$$

where θ is the angle between a and b . Larger values \Rightarrow vectors point in a similar direction (more related).

■ In Word2Vec:

$$s(w|c) = v_c \cdot u_w = \sum_{i=1}^d v_{c,i} u_{w,i}$$

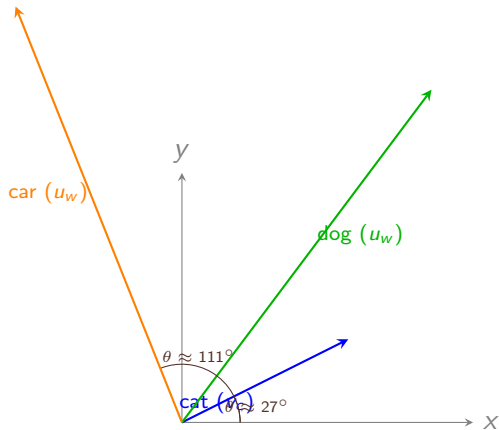
where v_c is the center word vector, u_w is a candidate context vector.

■ **Example:** $v_c = [2, 1]$ (“cat”), $u_w = [3, 4]$ (“dog”)

$$v_c \cdot u_w = (2 \times 3) + (1 \times 4) = 10$$

■ **Comparison:** $u_w = [-2, 5]$ (“car”)

$$v_c \cdot u_w = (2 \times -2) + (1 \times 5) = 1$$



- $v_c = [2, 1]$ (“cat”), $u_w = [3, 4]$ (“dog”) $v_c \cdot u_w = 10 \Rightarrow$ large positive (similar direction).
- $v_c = [2, 1]$ (“cat”), $u_w = [-2, 5]$ (“car”) $v_c \cdot u_w = 1 \Rightarrow$ small (weak relation).