Vector Semantics and Embeddings

Robert Minneker

2026-01-22

Sources

Content derived from: J&M Ch. 5

Part 0: Perplexity Review

Why do we need perplexity?

- How do we know if one language model is better than another?
- We need a metric that measures how well a model **predicts** real language
- Perplexity: “How surprised is the model by the test data?”

😌

Low Perplexity

"I expected that!"

Model assigns high probability

😳

High Perplexity

"That surprised me!"

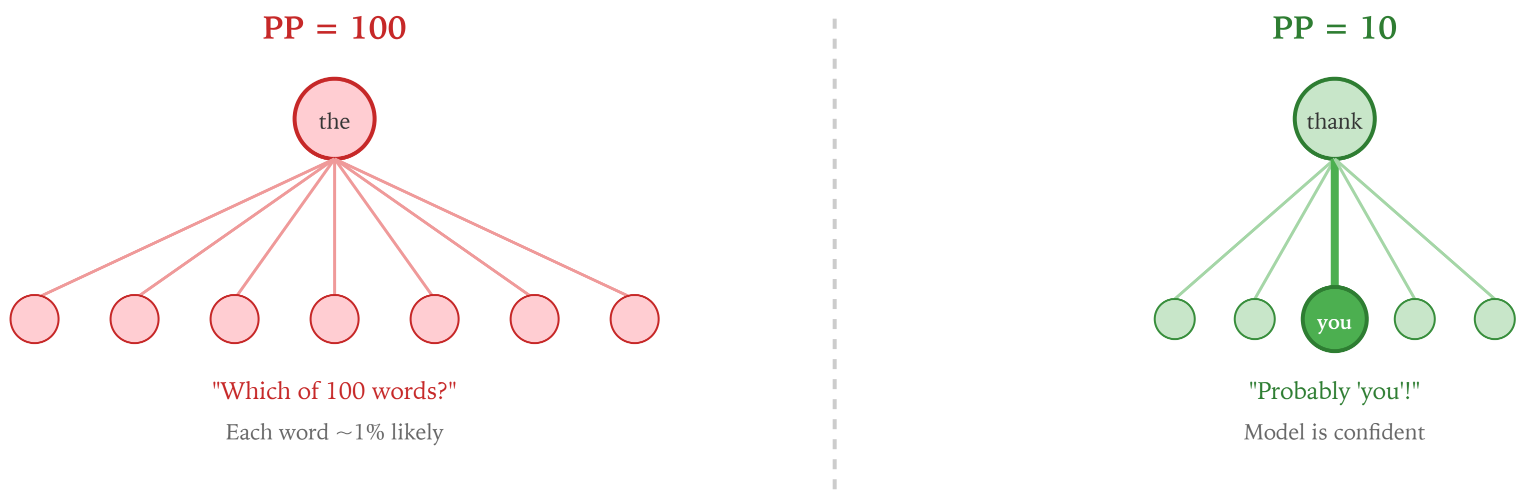
Model assigns low probability

Quick refresher: Types of means

Mean	Formula	When to use
Arithmetic	$\frac{1}{n}\sum_{i=1}^{n} x_i$	Averaging **additive** quantities (heights, test scores)
Geometric	$\sqrt[n]{\prod_{i=1}^{n} x_i}$	Averaging **multiplicative** quantities (probabilities) ← **perplexity!**
Harmonic	$\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$	Averaging **rates** (speeds, F1-score)
Quadratic (RMS)	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$	When **magnitude** matters more than sign (voltage, RMSE)

The core intuition: Branching factor

- Perplexity = "**effective number of choices**" at each word
- If model has perplexity 100 → as uncertain as choosing among 100 equally likely options



From probability to perplexity: The math

- For a test sequence $W = w_1, w_2, \ldots, w_N$:

Definition: Perplexity = geometric mean of inverse probabilities

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i \mid w_{i-1})}} = \left(\prod_{i=1}^{N} P(w_i \mid w_{i-1})\right)^{-\frac{1}{N}}$$

Key equivalence: This equals exponentiated cross-entropy!

$$\mathrm{PP}(W) = P(W)^{-\frac{1}{N}} = 2^{-\frac{1}{N}\log_2 P(W)} = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_{i-1})} = 2^{H(W)}$$

Why are these equivalent?

$$\mathrm{PP}(W) = \left(\prod_{i=1}^{N} P(w_i \mid w_{i-1})\right)^{-\frac{1}{N}} \qquad \text{(definition: geometric mean)}$$

$$= P(W)^{-\frac{1}{N}} \qquad \text{(chain rule: product } = P(W)\text{)}$$

$$= 2^{\log_2(P(W)^{-\frac{1}{N}})} \qquad \text{(identity: } x = 2^{\log_2 x}\text{)}$$

$$= 2^{-\frac{1}{N}\log_2 P(W)} \qquad \text{(log power rule)}$$

$$= 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_{i-1})} \qquad \text{(log of product = sum of logs)}$$

$$= \boxed{2^{H(W)}} \qquad \text{(definition of cross-entropy)}$$

Perplexity over a test sentence

For a single sentence $W = w_1 w_2 \cdots w_N$:

$$\mathrm{PP}(W) = P(w_1 w_2 \cdots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \cdots w_N)}}$$

Perplexity over a test corpus (in practice)

Problem: Multiplying thousands of probabilities → numerical underflow!

Solution: Work in log space — sum losses, normalize, exponentiate

$$\text{PP}(\text{corpus}) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid \text{context}_i)\right)$$

$$= \exp\left(\frac{1}{N}\sum_{i=1}^{N}\underbrace{-\log P(w_i \mid \text{context}_i)}_{\text{loss}_i}\right)$$

$$= \exp\left(\frac{1}{N}\sum_{i=1}^{N}\text{loss}_i\right) = \exp(\text{avg loss})$$

Algorithm: Keep running sum of losses → divide by N → exponentiate

Cross-entropy is your training loss!

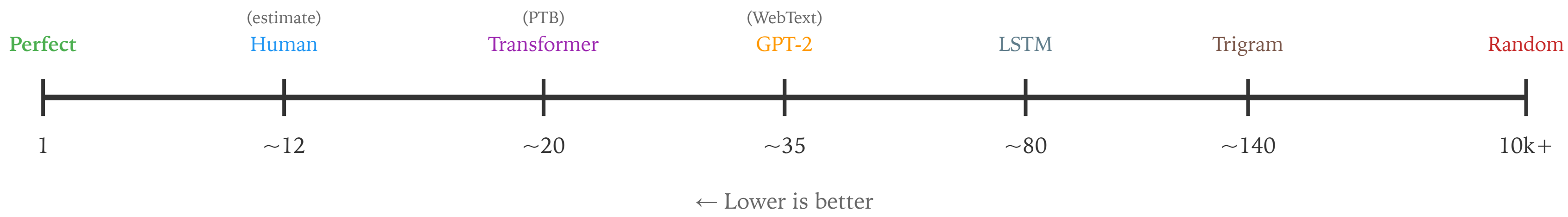
- In PyTorch, you're already optimizing perplexity:

```
# During training...
loss = F.cross_entropy(logits, targets) # This IS H(W)!

# To get perplexity:
perplexity = torch.exp(loss) # PP = e^H (natural log)
# or equivalently:
perplexity = 2 ** (loss / math.log(2)) # PP = 2^H (log
base 2)
```

Key insight: When you minimize cross-entropy loss, you're directly minimizing perplexity. Lower loss = lower perplexity = better language model.

What's a "good" perplexity? Benchmark context



Benchmark	Trigram	LSTM	Transformer
Penn Treebank	~140	~80	~20
WikiText-103	~150	~48	~18

Part 1: Foundations of Vector Semantics

“You shall know a word by the company it keeps” — Firth

- Words are characterized by their distributional properties, not in isolation
- “Bank” near “river” vs. “bank” near “money” reveals context-dependent meaning

bank

near: money, account, deposit, loan, interest

≠

bank

near: river, shore, water, fish, erosion

The distributional hypothesis

- Harris (1954): If two words appear in similar contexts, their meanings are likely similar

$$\text{contexts}(w_1) \approx \text{contexts}(w_2) \implies \text{meaning}(w_1) \approx \text{meaning}(w_2)$$

- No dictionary definitions needed—meaning emerges from usage patterns

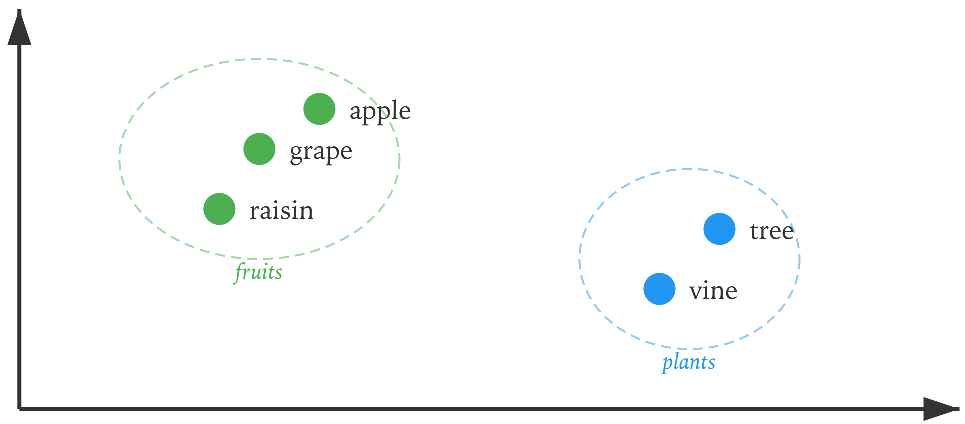
The antonyms problem: A limitation

- "Hot" and "cold" appear in **very similar contexts**:
  - "The water was ___"
  - "It's ___ outside today"
  - "___ temperature", "___ weather"
- But they have **opposite meanings**!

> **Key limitation:** Distributional similarity captures *topical relatedness*, not all aspects of meaning. Antonyms, hypernyms, and other semantic relations need additional modeling.

Each word maps to a vector in semantic space

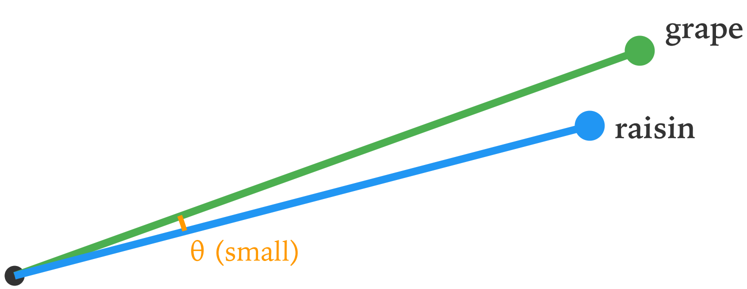
- Vocabulary V; each word $w \in V$ maps to $\mathbf{v}_w \in \mathbb{R}^d$
- Similar contexts → nearby vectors → similar meanings



Cosine similarity measures semantic closeness

$$\text{sim}(w_1, w_2) = \cos(\theta) = \frac{\mathbf{v}_{w_1} \cdot \mathbf{v}_{w_2}}{\|\mathbf{v}_{w_1}\| \|\mathbf{v}_{w_2}\|}$$

- **Direction** matters more than magnitude for meaning

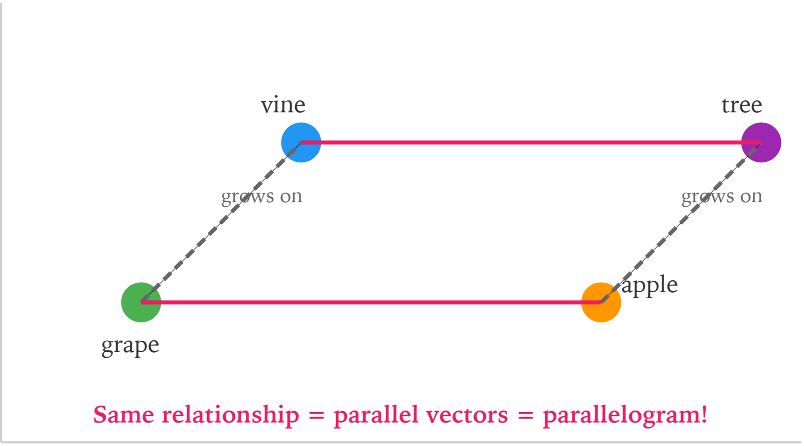


cos(θ) ≈ 0.95 → very similar

Vector arithmetic captures relationships

- Relationships encoded as **directions** in embedding space

$$\vec{v}_{\text{grape}} - \vec{v}_{\text{vine}} + \vec{v}_{\text{tree}} \approx \vec{v}_{\text{apple}}$$



The Conceptual Leap: Meaning Becomes Geometry

A Paradigm Shift in Linguistics

Traditional View

Meaning = symbolic definitions

"cat" defined by features:
+animate, +furry, +feline...

→

Vector Semantics

Meaning = position in space

"cat" = [0.2, -0.5, 0.8, ...]
300 learned dimensions

Why this matters:

- Semantic operations become **mathematical operations**
- Similarity → distance/angle; Analogy → vector arithmetic
- Meaning can be **computed**, not just looked up
- Enables **generalization** to unseen combinations

Part 2: Classical Count-Based Methods

TF-IDF: The Document Retrieval Baseline

- **Term Frequency (TF):** How often does word w appear in document d?
- **Inverse Document Frequency (IDF):** How rare is word w across all documents?

$$\text{TF-IDF}(w, d) = \text{TF}(w, d) \times \log \frac{|D|}{\text{DF}(w)}$$

High TF-IDF

"photosynthesis" in a biology paper
→ Frequent here, rare elsewhere
→ **Discriminative**

Low TF-IDF

"the" in any document
→ Frequent everywhere
→ **Not informative**

Limitation: TF-IDF captures document-level topicality, not fine-grained word similarity

Term-document matrices

- Matrix $\mathbf{X} \in \mathbb{R}^{|V| \times |D|}$ where X_{wd} counts word w in document d
- Each word = high-dimensional, sparse vector

	Doc 1 wine review	**Doc 2** botany text	**Doc 3** recipe
grape	15	8	3
wine	25	2	5
tree	0	18	1

"grape" vector: [15, 8, 3] → similar to "wine" [25, 2, 5]

Word-context matrices with sliding windows

- Matrix $\mathbf{C} \in \mathbb{R}^{|V|\times|V|}$: C_{wv} = count of v near w
- Window size controls what "near" means

```
The fresh grape juice tastes great
```

Window ±1:	fresh, juice	→ syntactic neighbors
Window ±5:	The, fresh, juice, tastes, great	→ topical neighbors

PMI: Measuring association strength

Pointwise Mutual Information: How much more often do words co-occur than expected?

$$\text{PMI}(w, c) = \log_2 \frac{P(w, c)}{P(w) \cdot P(c)}$$

Worked example

Corpus: 1000 word pairs

- "grape" appears in 20 pairs
- "wine" appears in 50 pairs
- "grape-wine" co-occurs 8 times

$P(\text{grape, wine}) = 8/1000 = 0.008$

$P(\text{grape}) \times P(\text{wine}) = 0.02 \times 0.05 = 0.001$

$\text{PMI} = \log_2(0.008/0.001) = \log_2(8) = \mathbf{3\ bits}$

Interpretation: 8× more likely than chance → strong association!

PPMI: Fixing negative infinity

Problem with PMI: If $P(w, c) = 0$, then PMI $= -\infty$

Solution: Positive PMI (PPMI) — clip negative values to zero

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

Why clip at zero?

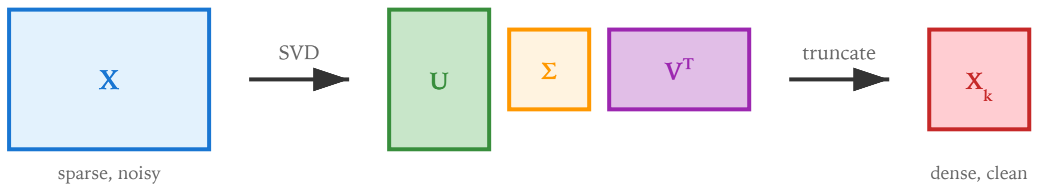
- Negative PMI often unreliable (sparse data)
- "Never co-occurred" ≠ "semantically opposite"
- Zeros are easier to handle (sparse matrices)

PPMI Matrix

- Sparse (mostly zeros)
- High-dimensional ($|V| \times |V|$)
- Better than raw counts
- Foundation for SVD/LSA

LSA: Dimensionality reduction reveals latent structure

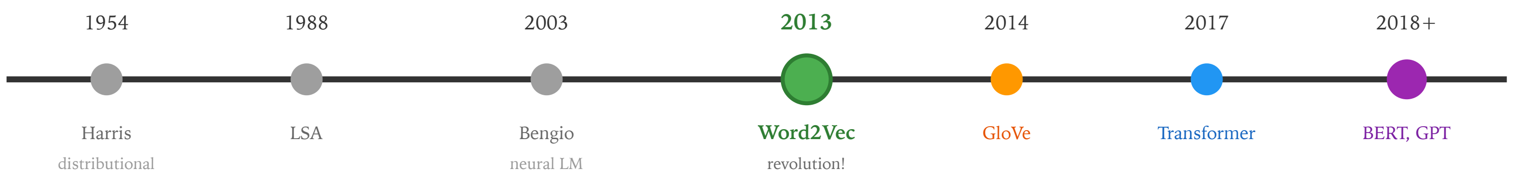
- SVD factorizes the matrix: $X = U\Sigma V^T$
- Truncate to k dimensions: keeps most important patterns



- "Grape" and "vineyard" become close even without direct co-occurrence
- Shared contexts ("wine", "harvest") create latent similarity

Part 3: Neural Embedding Methods

Historical timeline: The evolution of embeddings



Word2Vec: A Framework, Not a Single Algorithm

Word2Vec = Framework with Multiple Components

Architectures

- **SGNS**: Skip-gram + Negative Sampling
- **CBOW**: Continuous Bag of Words

Training Tricks

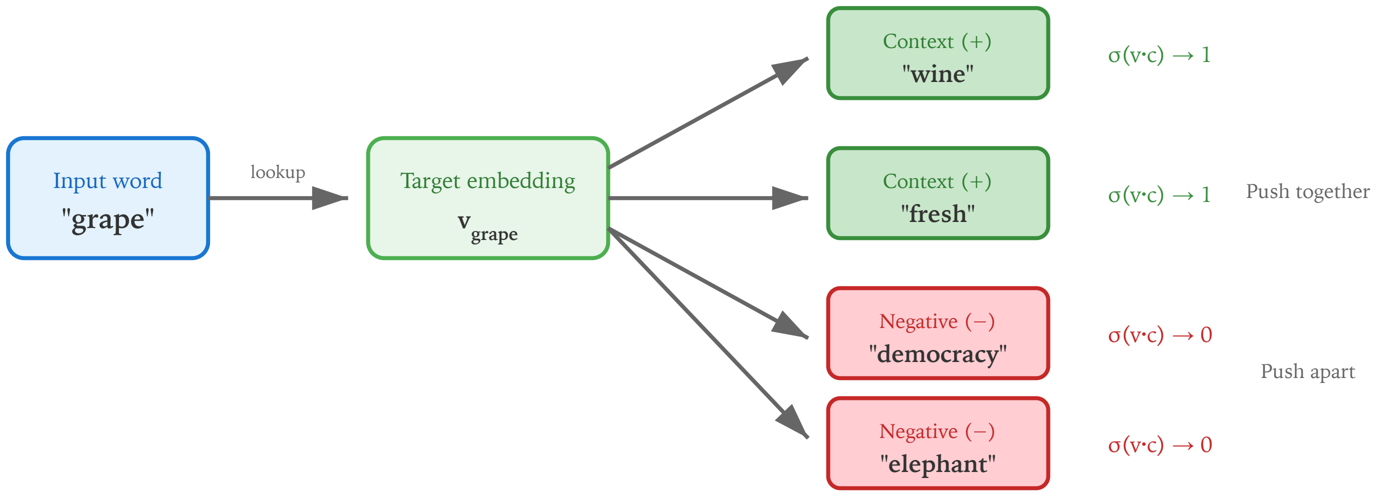
- Negative sampling
- Hierarchical softmax
- Subsampling frequent words

Key insight: Mikolov et al. (2013) introduced a **family** of methods, not one algorithm

- "Word2Vec" often refers to SGNS specifically (the most popular variant)
- Both architectures learn by **predicting** rather than counting

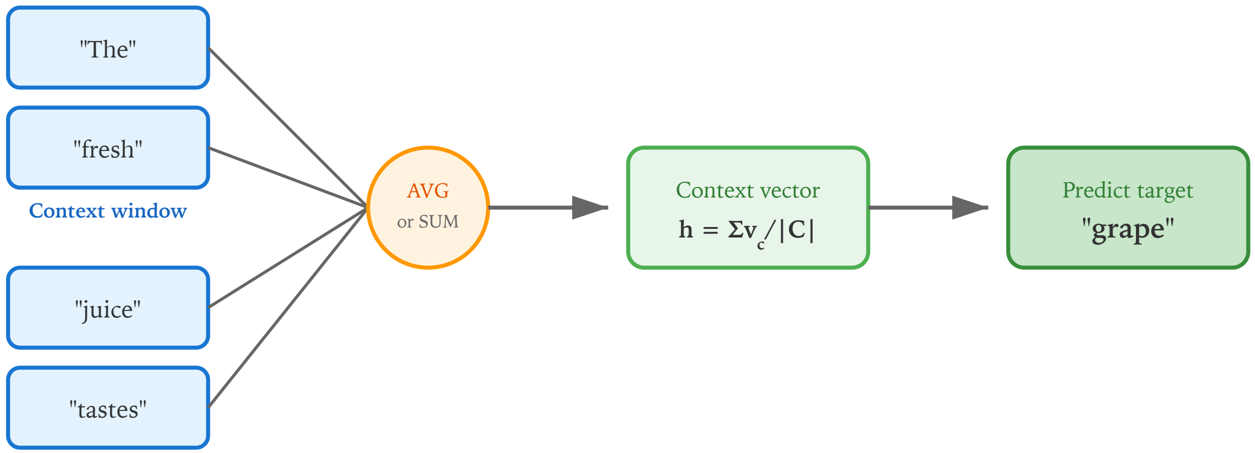
Skip-Gram with Negative Sampling (SGNS)

Core idea: Given a target word, predict its context words



Continuous Bag of Words (CBOW)

Core idea: Given context words, predict the target word



CBOW vs Skip-gram:

Aspect	CBOW	Skip-gram (SGNS)
Input	Context words	Target word
Output	Target word	Context words
Speed	Faster (1 prediction)	Slower (k predictions)
Rare words	Worse	Better

The Deep Connection: SGNS ≈ Implicit PMI Factorization

Levy & Goldberg (2014): A Landmark Discovery

$$\text{SGNS implicitly factorizes:} \quad \mathbf{W} \cdot \mathbf{W}'^T \approx \text{PMI}(w, c) - \log k$$

What this means:

- Skip-gram with negative sampling learns embeddings whose dot product approximates **shifted PMI**
- The "prediction" objective recovers the same statistical information as "counting"
- Neural and count-based methods are **two sides of the same coin**

Methods Comparison: The Full Picture

Method	What it captures	Matrix factorized	Training
TF-IDF	Document-level topicality	Weighted term-doc	Direct computation
PMI/PPMI + SVD	Word co-occurrence strength	PPMI matrix	Count → SVD
Word2Vec (SGNS)	Shifted PMI (implicitly)	$\text{PMI} - \log k$	SGD prediction
GloVe	Log co-occurrence (explicit)	$\log X$ weighted	SGD regression

GloVe: Explicit Global Optimization

GloVe's insight: If SGNS implicitly factorizes PMI, why not do it explicitly?

Word2Vec (SGNS)

- Learns by **predicting** context
- Stochastic: samples (word, context) pairs
- **Implicitly** factorizes PMI
- Online learning possible

GloVe

- Learns by **regression** on log-counts
- Batch: uses full co-occurrence matrix
- **Explicitly** minimizes reconstruction error
- Weighted by $f(X_{ij})$ to handle frequent pairs

Result: Similar embeddings, different training dynamics

Static vs contextualized embeddings

Static (Word2Vec, GloVe)

$f : V \to \mathbb{R}^d$

"The **bank** was steep"

→ [0.2, -0.5, …]

"The **bank** was closed"

→ [0.2, -0.5, …]

Same vector!

Contextualized (BERT)

$f : (w, C) \to \mathbb{R}^d$

"The **bank** was steep"

→ [0.3, 0.1, …]

"The **bank** was closed"

→ [-0.2, 0.4, …]

Different vectors!

Transformers: Self-attention for contextualization

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Every token attends to every other token
- Multiple layers refine representations
- Pre-training: Masked LM (BERT), autoregressive (GPT)

Concept Check

Why might word analogy tasks (grape - vine + tree = apple) work BETTER with static embeddings than contextualized ones?

Part 4: Evaluation

Similarity vs. relatedness: Know what you're measuring

Word Pair	WordSim-353 (relatedness)	SimLex-999 (similarity)
car - gasoline	HIGH	LOW
coffee - cup	HIGH	LOW
car - automobile	HIGH	HIGH

- **Relatedness** (WordSim): Are these words associated?
- **Similarity** (SimLex): Are these words interchangeable?

When analogies fail

- “king - man + woman = queen” is the famous success story
- But many analogies **don’t work**:

doctor - man + woman = ?	Often returns "nurse" (bias!)
Paris - France + Japan = ?	Sometimes "Tokyo", often noise
bigger - big + small = ?	Rarely returns "smaller"

Reality check: Google analogy dataset accuracy is ~60-75%, not 95%. Analogy arithmetic is a useful probe, not a reliable tool.

Extrinsic evaluation: The real test

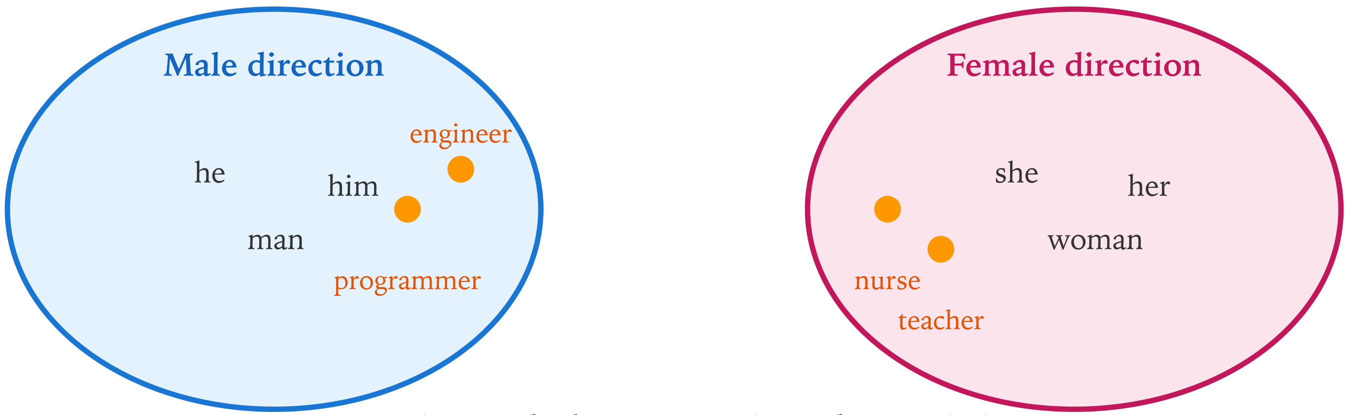
- **Intrinsic:** How good are the embeddings themselves?
- **Extrinsic:** How much do embeddings help downstream tasks?

Task	Without pretrained	With Word2Vec	With BERT
Sentiment	78%	84%	93%
NER	81%	88%	95%
Question Answering	65%	72%	89%

Part 5: Ethics and Bias

Embeddings encode societal biases

- Training data reflects historical biases
- Embeddings learn and **amplify** these patterns



Occupation words show systematic gender associations

WEAT: Measuring embedding bias

$$s(X, Y, A, B) = \frac{1}{|X|} \sum_{x \in X} s(x, A, B) - \frac{1}{|Y|} \sum_{y \in Y} s(y, A, B)$$

- Compare associations between word sets and attribute sets
- Parallels the psychological Implicit Association Test (IAT)

Example:
X = {programmer, engineer, scientist}
Y = {nurse, teacher, librarian}
A = {he, him, man}
B = {she, her, woman}

Finding: X more associated with A; Y more associated with B → gender bias

Debiasing: Partial solutions

Projection method

1. Identify "gender direction"
2. Project it out of all word vectors
3. "programmer" moves to neutral position

Limitations

- May hide rather than remove bias
- Some words "should" be gendered
- Bias can reappear in fine-tuning

Discussion

If occupation words like "engineer" are closer to "man" than "woman" in embedding space:

1. What downstream harms might this cause? (Think: hiring systems, search engines)
2. Can we ever create a truly "unbiased" language model?
3. Who should decide what counts as bias?

Summary

The Paradigm Shift
Meaning → Geometry
Semantic ops → Vector ops

Word2Vec = Framework
SGNS, CBOW architectures
Implicitly factorizes PMI

Methods Unified
TF-IDF → PMI → Word2Vec → GloVe
Count ↔ Prediction equivalence

Limitations & Ethics
Antonyms problem, analogy failures
Embeddings encode societal bias

Key takeaway: Vector semantics transforms meaning into geometry—powerful but imperfect.