

Prediction

Skip-gram with negative sampling, and what it is made of

James Pustejovsky

Brandeis University
Department of Computer Science

CS 135: Computational Semantics

Fall 2026

Where We Are

Session 8 reduced the co-occurrence matrix with the SVD: rotate, scale, rotate, and keep the largest singular values. It worked, and the demonstration at $k = 2$ showed both what it buys and how fragile the purchase is.

Today is the second route. Instead of building a matrix and factoring it, train a classifier on a task nobody cares about and keep the parameters.

Four questions to hold open

- 1 What is the model actually made of? Two lookup tables and one inner product.
- 2 Why gradient descent, when the SVD needed none?
- 3 What does it converge to – and does it really?
- 4 Analogies work. What does that tell us, and what does it not?

The last twenty minutes turn the fourth question into a structural one, and it is the question the rest of the course is built on.

The Second Route: Stop Counting, Start Predicting

Mikolov, Chen, Corrado & Dean (2013)

Do not build a matrix. Set up a **prediction task**, train a model with a small hidden layer, and take the learned parameters as the embeddings.

	Count-based	Prediction-based
Build	explicit $ V \times C $ matrix	nothing explicit
Then	factorize	optimize by SGD
Passes over corpus	one	several
New data	recompute	continue training
Vectors are	rows of $U_k \Sigma_k^\alpha$	learned parameters

The compression is now a *byproduct*: the hidden layer is narrow, so the model must find a d -dimensional code that supports the prediction. Nobody computes a rank- k approximation of anything – or so it appears.

Skip-Gram

Given a center word w , predict each context word c in a window around it.



Maximize, over the corpus:

$$\frac{1}{T} \sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w_{t+j} \mid w_t)$$

with the model

$$P(c \mid w) = \frac{\exp(\mathbf{c}_c \cdot \mathbf{w}_w)}{\sum_{c' \in V} \exp(\mathbf{c}_{c'} \cdot \mathbf{w}_w)}.$$

Note the two vectors per word: $\mathbf{w}_w$ when it is the center, $\mathbf{c}_c$ when it is the context. The next four slides are about that detail, which turns out not to be an implementation convenience.

The Denominator Is the Problem

$$P(c \mid w) = \frac{\exp(\mathbf{c}_c \cdot \mathbf{w}_w)}{\sum_{c' \in V} \exp(\mathbf{c}_{c'} \cdot \mathbf{w}_w)}$$

That sum runs over the entire vocabulary, for **every training example**. With $|V| = 10^5$ and 10^9 tokens in the corpus, each with a handful of context words, this is arithmetic on the order of 10^{14} dot products per epoch.

Two standard fixes:

- **Hierarchical softmax** – arrange the vocabulary as a binary tree, reducing the cost to $O(\log |V|)$ per example.
- **Negative sampling** – replace the whole probabilistic model with a different objective. **This is the one that matters for us.**

Notice the shape of the move: the softmax was the principled formulation, and it is abandoned for tractability. What replaces it is not an approximation to it.

Two Tables, Not One

The parameterization

For vocabulary V and dimension d , SGNS keeps **two** sets of vectors:

$$W \in \mathbb{R}^{|V| \times d} \quad (\text{word as } \textit{target}), \quad C \in \mathbb{R}^{|V| \times d} \quad (\text{word as } \textit{context})$$

Parameters: $\theta = \{W, C\}$, a total of $2|V|d$ numbers.

Every word gets two vectors, and which one is used depends on the role it is playing in the pair being scored:

$$\text{score}(w, c) = \mathbf{c}_c \cdot \mathbf{w}_w$$

the *target* vector of w against the *context* vector of c .

The usual justification is practical: a word rarely appears in its own context, so tying the tables would force $\mathbf{w}_w \cdot \mathbf{w}_w$ to be small, which is awkward. But something more interesting is going on.

An Asymmetry the Model Has and Discards

Part I gave likes the type $\langle e, \langle e, t \rangle \rangle$ and Alice the type e . The asymmetry between a functor and its argument was **in the type**, and it governed what could combine with what.

SGNS also has an asymmetry. A word occupies one of two positions, and gets a different vector in each.

	Part I	SGNS
The distinction	functor vs. argument	target vs. context
Where it lives	in the type	in which table you index
Enforced by	the type system	nothing
Survives training	yes	no – C is thrown away

Both vectors are elements of the same $\mathbb{R}^d$. Nothing stops you from putting a context vector where a target vector belongs, and the arithmetic will not complain. The asymmetry is a bookkeeping convention, not a type.

W and C Are U and V

Once you know the theorem, the two tables stop looking like an implementation detail.

$$WC^\top \approx M^{\text{PMI}} - \log k \quad \text{against} \quad U_k \Sigma_k V_k^\top = M_k$$

	SVD	SGNS
Word side	U_k , rows indexed by words	W
Context side	V_k , rows indexed by contexts	C
Scale	Σ_k , explicit and separate	absorbed into the two factors
Orthogonality	guaranteed	none

Both methods produce a two-sided factorization of a PMI-like matrix. The SVD hands you the scale in a third factor you can place where you like; SGNS has already distributed it, and you cannot recover the split.

What To Do With C

After training, the standard move is to keep W and discard C . Also common: use $W + C$.

Choice	SVD analogue	
keep W only	$W = U_k \Sigma_k^\alpha$, discard V	the usual default
use $W + C$	add the two sides	sometimes slightly better
concatenate $[W; C]$	keep both, $2d$ dimensions	rarely used

This is the α question from Session 8 in a different costume. There we asked where the scale should go once V is discarded; here we ask whether to discard C at all. In both cases the reconstruction $WC^\top$ is unchanged and only the *word geometry* moves.

And in both cases the answer is settled by downstream evaluation rather than by theory, which is the recurring situation in this part of the course.

Skip-Gram with Negative Sampling

Reframe. Instead of asking “which of $|V|$ words comes next,” ask a binary question: **was this (w, c) pair actually observed, or did I make it up?**

For each observed pair (w, c) , draw k fake context words $c_1, \dots, c_k$ from a noise distribution and maximize

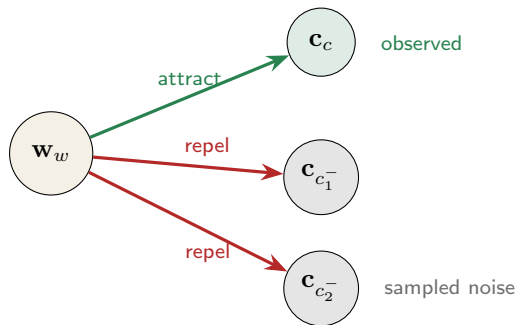
$$\log \sigma(\mathbf{c}_c \cdot \mathbf{w}_w) + \sum_{i=1}^k \mathbb{E}_{c_i \sim P_n} [\log \sigma(-\mathbf{c}_{c_i} \cdot \mathbf{w}_w)]$$

where $\sigma(x) = 1/(1 + e^{-x})$.

First term	push $\mathbf{w}_w$ and $\mathbf{c}_c$ together for pairs that occurred
Second term	push $\mathbf{w}_w$ and $\mathbf{c}_{c_i}$ apart for k pairs that did not

One $|V|$ -way classification has become $k + 1$ binary ones. Typical k : 5 to 20 for small corpora, 2 to 5 for large ones.

Attract and Repel



The first term is maximized by making $\mathbf{c}_c \cdot \mathbf{w}_w$ large, the second by making each $\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w$ small.

The word's position is determined by a tug of war between one thing it was seen with and k things it was not. There is no other force in the model.

The Noise Distribution: a Number You Have Seen

What is P_n , the distribution the fake context words are drawn from?

$$P_n(c) = \frac{\#(c)^{0.75}}{\sum_{c'} \#(c')^{0.75}}$$

That is the same 0.75 from Session 7.

Last session it appeared as *context distribution smoothing*, a fix for PMI overweighting rare contexts. Here it is the negative-sampling distribution, and Mikolov et al. report only that it “outperformed significantly” the alternatives tried.

Same exponent, same job

Both flatten the context distribution: rare contexts get a higher effective probability, frequent ones lower. In Session 7 that stopped barista beating cup on three observations; here it stops rare words being useless as negatives.

This is the first hint that the two methods are doing the same thing. It will not be the last.

Subsampling: the Third Solution to the

word2vec discards each token w during training with probability

$$P_{\text{drop}}(w) = 1 - \sqrt{\frac{t}{f(w)}}, \quad t \approx 10^{-5}$$

where $f(w)$ is the word's corpus frequency. Frequent words are dropped aggressively; rare ones are kept.

Session 7, PMI	the gets weight ≈ 0 , by the formula
Session 7, $\alpha = 0.75$	rare contexts are damped, by an exponent
Today, subsampling	the is deleted from the corpus, probabilistically

Three lectures, three unrelated-looking mechanisms, one underlying problem: high-frequency words dominate the counts and carry no information.

A second effect is easy to miss: deleting tokens *widens* the effective window, since the window is counted after deletion. Part of word2vec's reported gain comes from this, not the objective.

CBOW, Briefly

The other architecture in the same paper: predict the center word *from* the sum of its context vectors.

	Skip-gram	CBOW
Predicts	context from center	center from context
Training examples	one per (center, context)	one per position
Speed	slower	faster
Rare words	better	worse
Small corpora	better	worse

Skip-gram with negative sampling became the default, and it is the variant with the clean theoretical analysis.

Note in passing what CBOW does to compose a context: it **sums** the vectors. Session 10 will have something to say about that choice.

Why There Are Gradients Here At All

The SVD needed no gradients. Eckart–Young hands you the optimal low-rank factorization in closed form, and you read it off the spectrum.

But look at what that theorem promises. It minimizes $\sum_{ij} (M_{ij} - X_{ij})^2$, in which **every cell counts equally** – a cell resting on three thousand observations and a cell resting on two.

What SGNS does instead

Each pair contributes in proportion to $\#(w, c)$, so frequent pairs are fitted harder. That single change puts the problem outside the reach of the spectrum: weighted low-rank approximation has no closed form and is NP-hard in general.

The gradient is not an architectural choice, and it is not what makes the method “neural”. It is what remains once you decide that not all cells deserve equal trust.

Srebro & Jaakkola 2003; Gillis & Glineur 2011. We come back to this.

Two Identities

Everything follows from the derivative of the log-sigmoid.

$$\frac{d}{dx} \log \sigma(x) = \frac{\sigma'(x)}{\sigma(x)} = \frac{\sigma(x)(1 - \sigma(x))}{\sigma(x)} = 1 - \sigma(x)$$

$$\frac{d}{dx} \log \sigma(-x) = -(1 - \sigma(-x)) = -\sigma(x)$$

Read them as *error signals*. The first is how far the model's estimate of a true pair falls short of 1; the second is how much probability it wrongly assigns to a false pair.

Both are bounded in $(0, 1)$, which is why SGNS trains stably without a learning-rate schedule doing much work.

The Gradient for the Target Vector

From the positive term:

$$\frac{\partial}{\partial \mathbf{w}_w} \log \sigma(\mathbf{c}_c \cdot \mathbf{w}_w) = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{c}_c$$

From the i th negative term:

$$\frac{\partial}{\partial \mathbf{w}_w} \log \sigma(-\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) = -\sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{c}_{c_i^-}$$

$$\frac{\partial \ell}{\partial \mathbf{w}_w} = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{c}_c - \sum_{i=1}^k \sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{c}_{c_i^-}$$

The target moves toward the observed context and away from each sampled one, each step scaled by how surprised the model currently is.

The Gradients for the Context Vectors

$$\frac{\partial \ell}{\partial \mathbf{c}_c} = [1 - \sigma(\mathbf{c}_c \cdot \mathbf{w}_w)] \mathbf{w}_w \qquad \frac{\partial \ell}{\partial \mathbf{c}_{c_i^-}} = -\sigma(\mathbf{c}_{c_i^-} \cdot \mathbf{w}_w) \mathbf{w}_w$$

Symmetric to the last slide, with the roles exchanged: the context vectors move along $\pm \mathbf{w}_w$.

How much gets touched

One step updates $\mathbf{w}_w$, $\mathbf{c}_c$, and the k negatives: $2 + k$ vectors out of $2|V|$. With $|V| = 10^6$ and $k = 5$, that is seven rows out of two million.

Cost per step is $O(dk)$, independent of $|V|$. This is the entire practical argument for SGNS over the SVD, and it is an argument about sparsity of *updates*, not about representations.

Every Update Is a Multiple of the Other Vector

Look at the four gradients together. Each one has the same shape:

$$\Delta \mathbf{w}_w \in \text{span}\{\mathbf{c}_c, \mathbf{c}_{c_1^-}, \dots\}, \quad \Delta \mathbf{c} = \text{scalar} \times \mathbf{w}_w$$

No vector is ever moved in a direction that is not already a vector in the other table, and the step size is always a function of an inner product.

The entire learning dynamics take place inside the inner product. The model has no access to any other operation on its vectors – no outer product, no component-wise interaction, nothing that could raise the order of the representation.

Hold on to this. At the end of the session we will ask what was given up by building a semantics on one operation, and the answer will not be a matter of interpretation – it is visible right here, in the update rule.

The Update Rules

For one pair (w, c) with negatives $c_1^-, \dots, c_k^-$ and learning rate η :

1. **Scores.** $s^+ = \mathbf{c}_c \cdot \mathbf{w}_w, \quad s_i^- = \mathbf{c}_{c_i^-} \cdot \mathbf{w}_w$

2. **Target.**

$$\mathbf{w}_w \leftarrow \mathbf{w}_w + \eta \left[(1 - \sigma(s^+)) \mathbf{c}_c - \sum_{i=1}^k \sigma(s_i^-) \mathbf{c}_{c_i^-} \right]$$

3. **Observed context.** $\mathbf{c}_c \leftarrow \mathbf{c}_c + \eta (1 - \sigma(s^+)) \mathbf{w}_w$

4. **Each sampled context.** $\mathbf{c}_{c_i^-} \leftarrow \mathbf{c}_{c_i^-} - \eta \sigma(s_i^-) \mathbf{w}_w$

Typical settings: $d = 100$ – 300 , $L = 5$ – 10 , $k = 5$ – 20 on small corpora and 2 – 5 on large ones, η starting at 0.025 and decaying linearly.

Worked Example: the Setup

$d = 3$, target $w = \text{coffee}$, observed context $c = \text{cup}$, one negative $c^- = \text{elephant}$, $\eta = 0.1$.

$$\mathbf{w}_{\text{coffee}} = \begin{pmatrix} 0.5 \\ -0.2 \\ 0.3 \end{pmatrix}, \quad \mathbf{c}_{\text{cup}} = \begin{pmatrix} 0.4 \\ 0.1 \\ 0.6 \end{pmatrix}, \quad \mathbf{c}_{\text{elephant}} = \begin{pmatrix} -0.1 \\ 0.8 \\ -0.3 \end{pmatrix}$$

Scores.

$$s^+ = (0.4)(0.5) + (0.1)(-0.2) + (0.6)(0.3) = 0.360$$

$$s^- = (-0.1)(0.5) + (0.8)(-0.2) + (-0.3)(0.3) = -0.300$$

Error signals. $1 - \sigma(0.360) = 1 - 0.5890 = 0.4110$, $\sigma(-0.300) = 0.4256$

Worked Example: the Step

$$\Delta \mathbf{w} = 0.1 [\mathbf{0.4110} \mathbf{c}_{\text{cup}} - \mathbf{0.4256} \mathbf{c}_{\text{elephant}}] = \begin{pmatrix} 0.0207 \\ -0.0299 \\ 0.0374 \end{pmatrix}$$

	before	after
$\mathbf{w}_{\text{coffee}}$	(0.500, -0.200, 0.300)	(0.521, -0.230, 0.337)
$\mathbf{c}_{\text{cup}}$	(0.400, 0.100, 0.600)	(0.421, 0.092, 0.612)
$\mathbf{c}_{\text{elephant}}$	(-0.100, 0.800, -0.300)	(-0.121, 0.809, -0.313)

	before	after	change
$\mathbf{w}_{\text{coffee}} \cdot \mathbf{c}_{\text{cup}}$	0.3600	0.4045	+0.0445
$\mathbf{w}_{\text{coffee}} \cdot \mathbf{c}_{\text{elephant}}$	-0.3000	-0.3546	-0.0546
$\cos(\text{coffee, cup})$	0.8022	0.8167	+0.0145
$\cos(\text{coffee, elephant})$	-0.5657	-0.6122	-0.0465

What Is SGNS Actually Computing?

By 2014 the picture in the literature was roughly: counting is the old way, prediction is the new way, and prediction wins.

Levy & Goldberg (2014), *Neural Word Embedding as Implicit Matrix Factorization*

SGNS with k negative samples, at its optimum, satisfies

$$\mathbf{w}_w \cdot \mathbf{c}_c = \text{PMI}(w, c) - \log k$$

for every pair (w, c) . That is: **SGNS is factorizing the shifted PMI matrix.**

The same PMI you computed by hand last session, shifted by a constant that depends only on the number of negative samples.

The derivation is short. It is worth doing, because it is one of very few places in Part II where a claim about a neural method can be settled by algebra rather than by benchmark.

The Derivation, Part 1: Set Up

Write $x = \mathbf{w}_w \cdot \mathbf{c}_c$ and let $\#(w, c)$, $\#(w)$, $\#(c)$, N be the counts from Session 7.

If d is large enough, the vectors are expressive enough that each x can be chosen **independently** of the others. So the global objective decomposes, and we can optimize one cell at a time.

The objective contributed by the pair (w, c) :

$$\ell(x) = \#(w, c) \log \sigma(x) + k \cdot \frac{\#(w) \#(c)}{N} \log \sigma(-x).$$

$\#(w, c)$	how often the pair was observed – the positive term
$k \#(w) \#(c) / N$	how often it is expected as a negative sample

The second expression is the expected number of times (w, c) is drawn as a fake pair: w appears $\#(w)$ times, each draws k negatives, and c is drawn with probability $\#(c)/N$.

The Derivation, Part 2: Differentiate

Using $\frac{d}{dx} \log \sigma(x) = \sigma(-x)$ and $\frac{d}{dx} \log \sigma(-x) = -\sigma(x)$:

$$\frac{d\ell}{dx} = \#(w, c) \sigma(-x) - k \frac{\#(w) \#(c)}{N} \sigma(x) = 0.$$

Divide by $\sigma(x)$ and use $\sigma(-x)/\sigma(x) = e^{-x}$:

$$\#(w, c) e^{-x} = k \frac{\#(w) \#(c)}{N} \implies e^x = \frac{\#(w, c) N}{\#(w) \#(c)} \cdot \frac{1}{k}.$$

Take logs:

$$x = \log \left(\frac{\#(w, c) \cdot N}{\#(w) \cdot \#(c)} \right) - \log k = \boxed{\text{PMI}(w, c) - \log k}$$

Four lines. The quantity SGNS drives each dot product toward is the PMI of the pair, offset by $\log k$.

Logarithms here are natural, inherited from the sigmoid. Session 7 used $\log_2$, so this PMI is the earlier one times $\ln 2$ – a global constant that changes no cell's sign and no similarity ranking.

Reading the Result

Observation	Consequence
The target is $\text{PMI} - \log k$	SGNS is a count-based method
k shifts every cell by a constant	more negatives $\Rightarrow$ more cells pushed below zero
Cells with $\#(w, c) = 0$	never updated; effectively omitted, not set to $-\infty$
$d < V $	exact solution unreachable $\Rightarrow$ implicit low-rank factorization

Look at the second and third rows together. Shifting by $-\log k$ and then never visiting the zero cells has almost exactly the effect of **clipping negatives** – which is PPMI, arrived at by an entirely different route.

The two routes, aligned

Route 1: build PPMI, truncate the SVD. **Route 2:** optimize SGNS.

Route 2 solves a low-rank factorization of a shifted PMI matrix, by stochastic gradient descent instead of by SVD, with a different implicit loss.

The Theorem, and Whether to Believe It

Session 8 derived

$$\mathbf{c}_c \cdot \mathbf{w}_w = \text{PMI}(w, c) - \log k$$

from a first-order condition on a single pair, holding everything else fixed.

The derivation assumes	but in practice
each dot product is free to take any value	$d \ll V $ constrains them jointly
the optimum is reached	SGD stops early
P_n is the unigram, $\alpha = 1$	$\alpha = 0.75$

So the identity is a statement about an idealization. It is worth asking how closely the idealization is met – and that is something we can just measure.

The Experiment

Corpus	three topics, Zipfian word frequencies, 15% cross-topic leakage
Vocabulary	$ V = 15$
Window	$L = 2$, giving 180,132 observed (w, c) tokens
Model	$d = 12$, $k = 5$, noise $\alpha = 1$
Trained by	gradient ascent on the expected objective

The leakage matters. Without it every within-topic pair has nearly the same PMI, and there is no range over which to test anything. With it, $\text{PMI} - \log k$ spans $[-3.12, -0.59]$.

Then compare, for all 225 observed pairs, the learned $\mathbf{c}_c \cdot \mathbf{w}_w$ against the $\text{PMI}(w, c) - \log k$ computed directly from the counts.

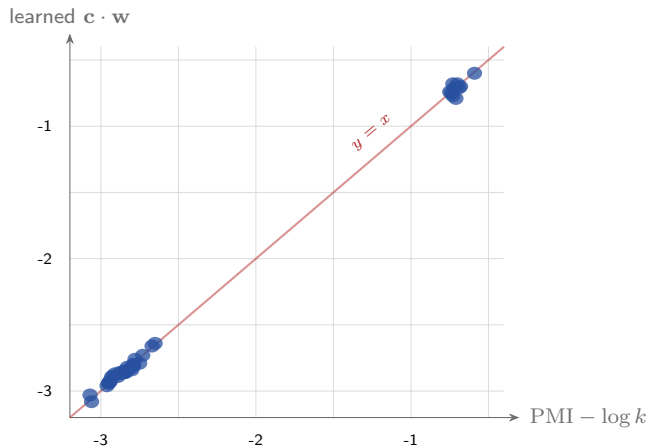
The Result

correlation	0.9996	
mean absolute error	0.0203	against a range of 2.52
best-fit slope	0.9990	theory says 1
best-fit intercept	-0.0010	theory says 0

pair	$\#(w, c)$	PMI	$\text{PMI} - \log k$	learned
coffee / cup	3093	0.863	-0.746	-0.691
coffee / brew	2384	0.898	-0.711	-0.657
river / water	4371	0.903	-0.707	-0.794
laptop / screen	4382	0.886	-0.724	-0.773
coffee / laptop	990	-1.232	-2.841	-2.837
river / battery	345	-1.274	-2.884	-2.911
coffee / river	930	-1.280	-2.890	-2.837

A quantity nobody computed, and nobody put in the loss, recovered to two decimal places.

The Same Thing, as a Picture



The two clusters are within-topic pairs (upper right) and cross-topic pairs (lower left). Both sit on the diagonal.

What the Fit Is Telling You

Slope 0.999	the relation is one-to-one, not merely monotone. SGNS is not learning “something correlated with PMI”; it is learning PMI
Intercept -0.001	the $-\log k$ offset is exactly right, which confirms that k enters as a constant shift and nothing else
Error 0.02	what is left is the finite d and finite optimization, and it is small

Raise k and every point slides down the diagonal by $\log k$ without changing shape. Lower d and the points scatter around the line, because the factorization can no longer fit every cell independently – which is the low-rank approximation reappearing.

The word “implicit” in “implicit matrix factorization” is doing no hedging. The matrix is there, to two decimal places.

So What Actually Changed in 2013?

Levy, Goldberg & Dagan (2015) tuned the count-based pipeline with word2vec's hyperparameters – context distribution smoothing, shifted PPML, subsampling, the α exponent – and compared fairly.

Finding	Detail
No consistent winner	tuned PPML+SVD matches SGNS on most similarity and analogy benchmarks
Hyperparameters dominate	the gap between settings exceeds the gap between methods
The transferable part	word2vec's <i>hyperparameters</i> , not its architecture, carried much of the gain

What did change: speed, memory, streaming rather than materializing a matrix, and an implementation anyone could run. Those are substantial, and they are engineering.

What did not change is the theory of meaning. Both routes represent a word by its weighted co-occurrence profile and compare by inner product.

GloVe: the Third Route Says So Out Loud

Pennington, Socher & Manning (2014)

Fit the log co-occurrence count directly:

$$\min \sum_{w,c} f(\#(w,c)) \left(\mathbf{w}_w \cdot \tilde{\mathbf{v}}_c + b_w + \tilde{b}_c - \log \#(w,c) \right)^2$$

Explicitly count-based, explicitly a factorization, motivated by the observation that *ratios* of co-occurrence probabilities carry the structure.

The weighting function

$$f(x) = \min \left((x/x_{\max})^{0.75}, 1 \right)$$

downweights rare pairs and caps frequent ones.

The exponent is 0.75 again – a third appearance, in a third role.

And note what f is for: it repairs the Frobenius-norm loss, which weights every cell equally. GloVe replaces “best in Frobenius norm” with “best in a loss that knows some cells are better estimated than others” – the qualification flagged on the Eckart–Young slide.

Three Routes, One Table

	SVD on PPMI	SGNS	GloVe
Matrix	explicit	implicit	explicit
Target quantity	$\text{PPMI}(w, c)$	$\text{PMI}(w, c) - \log k$	$\log \#(w, c)$
Loss	Frobenius	logistic	weighted least squares
Cell weighting	uniform	by frequency	$f(x)$, capped
Zero cells	clipped to 0	omitted	omitted
Solved by	eigendecomposition	SGD	SGD
Streaming	no	yes	no (needs counts first)

Read across any row: these are variations on *how to weight and how to optimize*, not variations on what a word is.

All three produce $\mathbf{w}_w \in \mathbb{R}^d$ such that $\mathbf{w}_w \cdot \mathbf{c}_c$ approximates a monotone function of the PMI of the pair. That is the destination the three routes share.

The Gains, Stated Precisely

Gain	Mechanism
Generalization	espresso/cappuccino: 0.000 $\rightarrow$ 1.000 via a shared latent direction
Compactness	$ C \approx 10^5$ becomes $d \approx 300$
Usability	a fixed-width real feature vector, which every downstream model wants
Differentiability	embeddings can be a <i>layer</i> , trained end to end with the task
Streaming	SGNS updates incrementally; no matrix to re-build

The fourth row has the largest downstream consequences, and is not about semantics at all: once a word representation is a differentiable parameter, it stops being preprocessing and becomes part of the model.

That is the road to Session 9: if the embedding is a layer, nothing requires it to be a lookup table.

Cosine, Not Dot Product

Training drives $\mathbf{c} \cdot \mathbf{w}$ toward PMI. Evaluation almost never uses the dot product.

$$\cos(\mathbf{w}_a, \mathbf{w}_b) = \frac{\mathbf{w}_a \cdot \mathbf{w}_b}{\|\mathbf{w}_a\| \|\mathbf{w}_b\|}$$

Dot product	grows with vector norm, and norm grows with frequency
Cosine	measures direction only

Note the small inconsistency this introduces. The objective optimizes a dot product; the evaluation reads an angle. Nothing guarantees that a model good at the first is best at the second, and the norm it discards carries real information – roughly, how confidently the word was placed.

This is a gap between what is optimized and what is measured, and it is rarely remarked on.

The Analogy Result

The finding that did more than any other to persuade people that embeddings encode structure:

$$\mathbf{w}_{\text{king}} - \mathbf{w}_{\text{man}} + \mathbf{w}_{\text{woman}} \approx \mathbf{w}_{\text{queen}}$$

and similarly for *Paris: France:: Rome: Italy, walking: walked:: swimming: swam*.

The apparent implication: relations are **directions**, and the space has linear structure that nobody put there.

If true in the strong form, this would be significant for us – it would suggest that some relational structure survives in the geometry, which is exactly the kind of thing Part IV wants.

So it is worth examining what the result actually shows.

The Analogy Result, Examined

The standard evaluation computes

$$\arg \max_{b^* \in V \setminus \{a, a^*, b\}} \cos(\mathbf{w}_{b^*}, \mathbf{w}_{a^*} - \mathbf{w}_a + \mathbf{w}_b)$$

Finding	Source
The three query words are <i>excluded</i> from the answer set	standard in every implementation
Without that exclusion, the top answer is usually b itself	Linzen (2016)
A baseline that just returns the nearest neighbor of b scores well	Linzen (2016)
$\cos$ multiplication (3CosMul) beats addition	Levy & Goldberg (2014)
Performance varies sharply by relation type	several

Much of the effect is neighborhood structure plus a rule forbidding the obvious wrong answers. The arithmetic contributes, but less than the headline suggests and unevenly across relations. There is real linear structure for some relations – morphological and geographic especially – and the demonstration overstates how much of the work the arithmetic does.

Why Analogies Follow From the Theorem

The analogy $a : b :: c : d$ holds when the PMI differences match across every context:

$$\text{PMI}(a, c') - \text{PMI}(b, c') \approx \text{PMI}(c, c') - \text{PMI}(d, c') \quad \text{for all } c'$$

Substituting $\mathbf{w}_w \cdot \mathbf{c}_{c'} \approx \text{PMI}(w, c') - \log k$, the $\log k$ cancels in each difference:

$$(\mathbf{w}_a - \mathbf{w}_b) \cdot \mathbf{c}_{c'} \approx (\mathbf{w}_c - \mathbf{w}_d) \cdot \mathbf{c}_{c'} \quad \text{for all } c'$$

If that holds against enough context vectors to span the space, then

$$\mathbf{w}_a - \mathbf{w}_b \approx \mathbf{w}_c - \mathbf{w}_d$$

So the parallelogram is not an emergent mystery. It is what a PMI-ratio regularity in the corpus looks like after you take logarithms, and logarithms turn ratios into differences.

What the Parallelogram Commits You To

Read the claim structurally. If a relation R is a fixed vector $\mathbf{r}$ with $\mathbf{w}_{R(x)} = \mathbf{w}_x + \mathbf{r}$, then:

Property of vector addition	Property it forces on the relation
$\mathbf{r} + \mathbf{s} = \mathbf{s} + \mathbf{r}$	relations commute
$\mathbf{r} + (-\mathbf{r}) = \mathbf{0}$	every relation is invertible
$\mathbf{r}$ does not depend on $\mathbf{x}$	the relation applies uniformly

Meanings under relations form an abelian group. That is a strong claim, and it is why the successes are what they are: plural, comparative, past tense, capital-of, currency-of are all near-invertible and near-uniform.

And it is why *cause*, *part-of*, *entails* do not work this way. Those relations are not invertible, and some are not even symmetric in the relevant sense.

The Same Commutativity, Twice

Analogies

vector addition is commutative, and that is exactly what makes the parallelogram work

Composition

vector addition is commutative, and that is exactly what makes *dog bites man* and *man bites dog* identical

One property, celebrated in one place and fatal in the other. The analogy result is not evidence that the space is compositional. It is evidence that the space is **additive**, and additivity is the specific thing that composition cannot be.

Keep that in view for the last two slides. The analogy result is the best case the inner-product world can make, and it is a case about relations between *single words* – never about combining two of them into a third.

The Window Chooses What “Similar” Means

Small window ($L \leq 3$)	Large window ($L \geq 8$)
syntactic similarity	topical association
dog $\sim$ cat, dogs	dog $\sim$ pet, veterinarian
tight substitutability	broad relatedness

Word2vec also samples the window size uniformly from $\{1, \dots, L\}$ at each position, which weights nearer words more heavily without any explicit weighting term.

This is the Session 7 knob, unchanged. The definition of “context” was always the place where a linguistic decision entered the pipeline, and it still is – it is simply now a hyperparameter rather than a stated choice.

Dependency-based contexts sharpen the effect further, and are the closest this tradition comes to the Word Sketch idea.

What Was Lost

Loss	Detail
Interpretable coordinates	a PPMI column was cup; dimension 137 is nothing
Sparsity	every word now has a nonzero value in every dimension
Auditability	you can check a PPMI cell against the corpus; you cannot check a trained weight
The exact regime	no coordinate per entity, so 6B's encoding is now unreachable

The last row closes a loop opened in Session 6B, whose “Changing the Base Case” slide replaced the one-hot $\mathbf{e}_i$ with a dense learned $\mathbf{w}_i$. Session 7 gave us non-orthogonality but kept a coordinate per context.

Today the coordinates are gone: no basis vector corresponds to anything, not an entity and not a context. 6B's approximate regime has arrived in full – $d \ll n$, dense, learned, and no way back.

One Loss Worth Dwelling On: Antonyms Got Worse

Session 7 established that hot and cold are distributionally near-identical, and argued this was a principled consequence rather than a bug.

Reduction **amplifies** it. The mechanism is the one we demonstrated:

- Low-rank approximation pulls each word toward words with similar profiles.
- hot and cold have similar profiles by construction.
- So they are pulled toward each other, and the small idiosyncratic differences that distinguished them are exactly the low-variance directions that truncation discards.

The distinguishing information lives in the tail of the spectrum, and the tail is what we threw away.

This is the same structure as the $k = 3$ result earlier, running the other direction: truncation is a choice about which distinctions to preserve, and it has no way of knowing that antonymy matters and topic does not.

Embeddings Encode the Corpus's Associations

Study	Finding
Bolukbasi et al. (2016)	<i>man</i> : <i>computer</i> <i>program-</i> <i>mer</i> :: <i>woman</i> : <i>homemaker</i> recovered by the analogy method
Caliskan, Bryson & Narayanan (2017)	embeddings reproduce human implicit- association (IAT) effects, including ones tracking documented social bias

The structure of the argument is identical to the antonym argument: (1) the method represents a word by the company it keeps; (2) in the corpus, these words keep that company; (3) therefore the representation encodes the association.

This is the distributional hypothesis working as specified, on a corpus that reflects the world that produced it – not an algorithmic defect, and not fixed by a better optimizer.

Gonen & Goldberg (2019) showed that standard debiasing projections remove the *direction* while leaving the clustering intact.

The Same Geometry, Read as Bias

Note that the Bolukbasi result *is* the analogy result. The same displacement structure that gives king–queen gives programmer–homemaker, for the same reason: the PMI ratios in the corpus have that shape.

So the natural fix is geometric. Identify a “gender direction” and project it out of every vector.

Gonen & Goldberg (2019)

After projection, the affected words remain clustered together. A classifier trained to recover the original grouping still succeeds. The projection removed the *measurement* and left the structure in place.

The association was never confined to one direction. It is distributed through the representation, because it is distributed through the corpus – and a rank-one intervention cannot reach it.

The Ledger, Unchanged

	Part I	Session 7	Today
Similarity	—	✓	✓✓
Generalizes to unseen pairs	—	—	✓
Compact	—	—	✓
Differentiable	—	—	✓
Antonyms distinguished	✓	—	—
Negation	✓	—	—
Quantifier scope	✓	—	—
Monotonicity	✓	—	—
Composition	✓	—	— Session 10

The top block improved. The bottom block is identical to Session 7's.

Reduction is a better *lexical* representation. It is not a step toward compositional semantics, and three hundred dimensions of it are no closer than fifty thousand were.

The Operation Nobody Chose

Look back at the update rule from the middle of this session:

$$\mathbf{w}_w \leftarrow \mathbf{w}_w + \eta \left[(1 - \sigma(s^+)) \mathbf{c}_c - \sum_i \sigma(s_i^-) \mathbf{c}_{c_i^-} \right]$$

Every vector moves along another vector, by an amount computed from an inner product. There is no step anywhere in the training loop that is not one of those two things.

	result	what happens to the index
inner $\mathbf{a} \cdot \mathbf{b}$	a scalar	summed away
outer $\mathbf{a}\mathbf{b}^\top$	a matrix	kept; order rises from 1 to 2

Distributional semantics chose the first row, and never argued for it. The inner product is the operation that throws away *which dimension agreed with which* – which is exactly the information composition would need.

Four Positions

	operation	keeps	costs
Distributional, incl. SGNS	$\mathbf{a} \cdot \mathbf{b}$	similarity	all structure
Tensor products	$\mathbf{a}\mathbf{b}^\top$	structure, exactly	dimension, exponent
Vector symbolic architectures	a folded outer product	structure, fixed dimension	exactness
Geometric algebra	$\mathbf{a}\mathbf{b} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b}$	both, as grades	you must track grades

Today's deck established the first row in the strongest available way. Not by interpreting the model – by reading its gradients.

The analogy result is the best case that row can make, and we saw what it rests on: additivity. That is precisely the property composition cannot have, which is why *dog bites man* and *man bites dog* are still the same point.

Next session takes the remaining three rows seriously.

Practice

1. A word's target vector $\mathbf{w}_w$ and its context vector $\mathbf{c}_w$ are both learned. Would you expect $\cos(\mathbf{w}_w, \mathbf{c}_w)$ to be high or low, and why? What does your answer imply about using $W + C$?
2. In the SGNS derivation we assumed each $\mathbf{w}_w \cdot \mathbf{c}_c$ could be set independently. For what d is that exactly true, and what goes wrong below it?
3. SGNS drives $\mathbf{w}_w \cdot \mathbf{c}_c \rightarrow \text{PMI}(w, c) - \log k$. What value of k makes the target equal to PMI itself? What happens to the sparsity of the implied matrix as k grows?
4. Take the generalization corpus. Add one occurrence of espresso with foam. Predict – before computing – whether the $k = 3$ cosine for espresso/cappuccino rises above zero, and say why.
5. **Harder.** CBOW composes a context by summing vectors. Give two English sentence pairs that CBOW's context representation cannot distinguish, and say which property of addition is responsible.

Exercise 5 is a rehearsal for Session 10. Keep your answer.

Exercises

- 1 In the worked example, $\cos(\text{coffee}, \text{elephant})$ moved by -0.047 while $\cos(\text{coffee}, \text{cup})$ moved by $+0.015$. The learning rate was the same for both. Explain the asymmetry from the two error signals.
- 2 The verification used noise exponent $\alpha = 1$. Predict, with reasons, what happens to the slope and intercept of the fit when $\alpha = 0.75$, and say which quantity the learned dot products would then be tracking.
- 3 Suppose you keep $W + C$ rather than W . Show that the reconstruction WC^T is unaffected by this choice, and explain why the word-to-word cosines are not.
- 4 **Harder.** The parallelogram makes relations an abelian group. Name a lexical relation that is genuinely non-invertible, predict what the analogy test would report for it, and say what a representation would need in order to do better.

Reading: Mikolov et al. (2013a, 2013b); Levy & Goldberg (2014); Levy, Goldberg & Dagan (2015); Bolukbasi et al. (2016); Gonen & Goldberg (2019).

Summary

Point	Detail
The second route	train a classifier on a pretext task and keep the parameters
The denominator	full softmax costs $O(V d)$ per step; negative sampling replaces it
Two tables	W for the target role, C for the context role; $2 V d$ parameters
The asymmetry	a functor/argument-like distinction, held in bookkeeping and then discarded
W and C are U and V	both routes are two-sided factorizations of a PMI-like matrix
What to do with C	the α question again: reconstruction fixed, word geometry not
The 0.75 exponent	the same number as context distribution smoothing, same job
Why gradients	weighting the cells forfeits Eckart–Young, and with it the spectrum
The four gradients	each moves a vector along another vector, scaled by an inner product
Cost	$O(dk)$ per step, independent of $ V $: $2 + k$ rows touched out of $2 V $
The theorem	$\mathbf{w}_w \cdot \mathbf{c}_c = \text{PMI}(w, c) - \log k$
Verified	correlation 0.9996, slope 0.999, intercept -0.001
Consequence	count vs. predict is a disagreement about which cells to trust
GloVe	explicit counts, weighted least squares; the 0.75 exponent a third time
Cosine vs. dot	the objective optimizes one and the evaluation reads the other
Analogies	follow from PMI ratios; logarithms turn ratios into differences
What that commits you to	relations as an abelian group: commutative, invertible, uniform
The same commutativity	celebrated in analogy, fatal in <i>dog bites man</i>
Antonyms, negation	unchanged; the evidence is distribution, and it cannot see them
Bias	Gonen & Goldberg: projection removes the measurement, not the structure
The operation	everything today ran on the inner product, which sums the index away

Reading

Reading: Lappin & Shanahan Ch. 4.

Primary sources:

- Mikolov, Chen, Corrado & Dean (2013a), *Efficient Estimation of Word Representations*
- Mikolov, Sutskever, Chen, Corrado & Dean (2013b), *Distributed Representations of Words and Phrases*
- Levy & Goldberg (2014), *Neural Word Embedding as Implicit Matrix Factorization* – the derivation is §2–3 and runs four pages
- Levy, Goldberg & Dagan (2015), *Improving Distributional Similarity with Lessons Learned from Word Embeddings*
- Pennington, Socher & Manning (2014), *GloVe*

On the analogy result and on bias:

- Linzen (2016), *Issues in Evaluating Semantic Spaces Using Word Analogies*
- Bolukbasi et al. (2016); Caliskan, Bryson & Narayanan (2017)
- Gonen & Goldberg (2019), *Lipstick on a Pig* – short, and worth reading beside Bolukbasi

On the weighting point from slide 15: Srebro & Jaakkola (2003); Gillis & Glineur (2011).

Questions?

Next: what a representation would need
in order to compose