

# Vector Symbolic Architectures

Structure without symbols, and where the symbols come back

James Pustejovsky

Brandeis University  
Department of Computer Science

CS 135: Computational Semantics

Fall 2026

# Roadmap

---

|             |                            |                                           |
|-------------|----------------------------|-------------------------------------------|
| <b>I</b>    | Where we left off          | the diagnosis, and the bet VSA makes      |
| <b>II</b>   | The alphabet               | why randomness gives you symbols          |
| <b>III</b>  | From linear algebra to MAP | which operations, why, and worked by hand |
| <b>IV</b>   | Three operations at scale  | bind, bundle, permute at $d = 10,000$     |
| <b>V</b>    | Cleanup memory             | where discreteness comes back             |
| <b>VI</b>   | Building structures        | records, sequences, events                |
| <b>VII</b>  | The payoff                 | analogy with no training at all           |
| <b>VIII</b> | Types for a VSA            | roles, values, operations, structures     |
| <b>IX</b>   | Assessment                 | what it buys, what it cannot do           |
| <b>X</b>    | Close                      | back to the question from last session    |

---

Last session ended with a diagnosis: binding by circular convolution destroys the antisymmetric half, so it cannot encode argument order. Today we see the architecture built on top of that operation, and how it copes.

Everything today is checkable arithmetic on vectors you could generate in three lines of numpy. There is no training anywhere in this deck.

## Where We Are



Sessions 7 through 9 learned vectors that carry meaning and cannot carry structure. Sessions 10 and 11 found out why: structure needs the antisymmetric half, and every operation we had discarded it.

### Today's move

Go to the opposite extreme. Give up on the vectors carrying meaning at all, and see how much structure you can build with pure algebra. The result is a system that can represent *John loves Mary*, answer questions about it, and do analogy – with no training of any kind.

Then, next session, the geometric product, which does not force the choice.

# The Diagnosis, Restated

Session 11 asked where symbolic reasoning resides in a vector representation, and located three capacities:

| Capacity                                             | Lives in               | Status after Session 11 |
|------------------------------------------------------|------------------------|-------------------------|
| identity: tell dog from cat                          | the symmetric half     | solved by dimension     |
| structure: tell loves( $j, m$ ) from loves( $m, j$ ) | the antisymmetric half | destroyed by the fold   |
| inference: compose, then decompose                   | both                   | only approximate        |

The outer product  $\mathbf{r}\mathbf{f}^\top$  keeps all three and costs  $n^2$ . Circular convolution costs  $n$  and keeps the first. We proved the fold annihilates the antisymmetric half exactly, so the loss is not a matter of precision.

**So the question for today.** Given a binding operation that cannot distinguish agent from theme, how do you build a system that represents *John loves Mary* and answers questions about it? You supply the missing order by other means, and the other means are the interesting part.

# The Bet

A vector symbolic architecture makes one wager and spends the whole design on it.

## The wager

Give up learning the representations. Assign every atomic symbol a *random* high-dimensional vector, chosen once and never adjusted. Then build all structure with fixed algebraic operations.

|                         | Distributional semantics   | VSA                          |
|-------------------------|----------------------------|------------------------------|
| where vectors come from | learned from corpora       | drawn at random              |
| what similarity means   | graded, meaningful         | near zero for distinct atoms |
| what composition is     | addition, or a learned net | fixed algebra                |
| what you get back       | a point in a space         | a decomposable structure     |

Notice how much is given up. dog and cat will be as dissimilar as dog and Tuesday. What you get in exchange is the thing distributional semantics could not do: put a structure in, take the structure back out.

# Why That Is the Whole Trick

We established last session that random vectors in high dimension are nearly orthogonal – mean cos of zero, spread  $1/\sqrt{d}$ , and at  $d = 10,000$  the worst of twenty thousand pairs reached only 0.039. *That one fact is doing three separate jobs here, and they are worth pulling apart.*

- 1 **A large alphabet, free.** You need no design, no training, and no coordination. Any process that can draw random numbers can mint new symbols, and they will not collide.
- 2 **Superposition without confusion.** Add  $\mathbf{a} + \mathbf{b} + \mathbf{c}$  and the sum is still measurably similar to each one and to nothing else. A set becomes a single vector.
- 3 **Robustness.** Flip a third of the coordinates and the cosine to the original is still about 0.33, far above the noise floor of 0.01. There is no fragile bit anywhere.

*Recall which half of a bilinear form this is.* Distinctness is  $\mathbf{x}^\top G \mathbf{y} = 0$ , a *symmetric* condition. So dimension buys you the identity capacity and nothing else. *Nothing in a random codebook says anything about order.*

“Hyperdimensional computing” names the same field from this fact: the representational power comes from  $d$ , not from cleverness in the code.

## Robustness, and Why Nothing Here Is Fragile

Take an atom at  $d = 10,000$  and corrupt it: flip a fraction  $f$  of its coordinates at random. The cosine to the original falls linearly, and the noise floor does not move.

| fraction flipped $f$          | 0    | 0.1  | 0.2  | 0.3  | 0.4  | 0.49 |
|-------------------------------|------|------|------|------|------|------|
| cos to the original, $1 - 2f$ | 1.00 | 0.80 | 0.60 | 0.40 | 0.20 | 0.02 |
| noise floor at this $d$       |      |      |      | 0.01 |      |      |

Flip 3,000 of the 10,000 coordinates and cleanup still identifies the atom without ambiguity, because 0.40 against a floor of 0.01 is not a close call.

**There is no most-significant bit.** Every coordinate contributes equally and none is load-bearing, which is the opposite of how a symbol is stored in a conventional machine, where one flipped bit in a pointer loses everything.

This is the property that interests the hardware community: a representation that tolerates faulty memory and analog noise by construction.

## What These Atoms Are Not

A point worth making before the machinery, because it is where intuitions from Sessions 7 to 9 will mislead you.

|                              | word2vec vector for dog | VSA atom for dog          |
|------------------------------|-------------------------|---------------------------|
| cos with cat                 | $\approx 0.8$           | $\approx 0.00$            |
| cos with Tuesday             | $\approx 0.1$           | $\approx 0.00$            |
| carries meaning?             | yes, in its direction   | no, only an identity      |
| replaceable by another draw? | no                      | yes, relabel and continue |

A VSA atom is a name, not a meaning. It is closer to a Lisp symbol or a Prolog constant than to an embedding.

Which raises the obvious objection: if the atoms carry no semantic similarity, in what sense is this *semantics*? Part IX. Hold the thought.

# Start From What Linear Algebra Already Gives You

Before any VSA vocabulary, here is the toolkit you already have for vectors in  $\mathbb{R}^d$ :

| Operation           | Written                                                                     | Output                     | Output similar to the inputs?   |
|---------------------|-----------------------------------------------------------------------------|----------------------------|---------------------------------|
| addition            | $\mathbf{a} + \mathbf{b}$                                                   | a vector in $\mathbb{R}^d$ | yes, to both                    |
| scaling             | $c \mathbf{a}$                                                              | a vector in $\mathbb{R}^d$ | yes: same direction             |
| dot product         | $\mathbf{a} \cdot \mathbf{b}$                                               | a number                   | – it <i>measures</i> similarity |
| matrix times vector | $M \mathbf{a}$                                                              | a vector in $\mathbb{R}^d$ | depends on $M$                  |
| elementwise product | $\mathbf{a} \odot \mathbf{b}$ , $(\mathbf{a} \odot \mathbf{b})_i = a_i b_i$ | a vector in $\mathbb{R}^d$ | no, for random $\pm 1$ vectors  |
| outer product       | $\mathbf{a} \mathbf{b}^\top$                                                | a $d \times d$ matrix      | – it leaves $\mathbb{R}^d$      |

A VSA uses nothing outside this table. Four rows keep you in  $\mathbb{R}^d$ , and they differ in the property in the last column: whether the result still resembles what went in.

Notation for the rest of the deck:  $\otimes$  names the VSA's binding, whatever computes it, and the outer product is always written  $\mathbf{a} \mathbf{b}^\top$ .

## What a Symbol System Needs From Them

A system that stores *John loves Mary* and answers questions about it has four jobs, and each job constrains the operation that does it.

| Job            | Example                      | The operation must                                             |
|----------------|------------------------------|----------------------------------------------------------------|
| <b>collect</b> | a sentence's words, as a set | stay similar to every part; ignore order                       |
| <b>pair</b>    | a role with its filler       | resemble neither part; be undoable; distribute over collecting |
| <b>mark</b>    | a position, or a depth       | be undoable; send its input somewhere new                      |
| <b>compare</b> | is this the atom john?       | give a graded score: the cosine                                |

One more condition covers all four: **every result must land back in  $\mathbb{R}^d$** , so that it can be collected, paired, or marked again. The outer product fails exactly this, with  $d^2$  numbers for one pair.

Why “distribute over collecting”? A record is a collection of pairs, and you want to undo one pair without taking the collection apart:  $\mathbf{r} \otimes (\mathbf{x} + \mathbf{y}) = \mathbf{r} \otimes \mathbf{x} + \mathbf{r} \otimes \mathbf{y}$ .

## Three Candidates for Pairing

All three are **bilinear** – linear in each argument separately – so distributing over collecting comes free. They differ everywhere else:

|                           | outer product $\mathbf{r}\mathbf{f}^\top$ | convolution $\mathbf{r} \circledast \mathbf{f}$ | elementwise $\mathbf{r} \odot \mathbf{f}$ |
|---------------------------|-------------------------------------------|-------------------------------------------------|-------------------------------------------|
| architecture              | TPR                                       | HRR                                             | MAP                                       |
| stays in $\mathbb{R}^d$ ? | no: $d^2$ numbers                         | yes                                             | yes                                       |
| resembles neither part?   | – another space                           | yes                                             | yes                                       |
| undoable?                 | exactly                                   | approximately                                   | exactly, if coordinates are $\pm 1$       |
| commutative?              | no                                        | yes                                             | yes                                       |
| cost per pair             | $d^2$                                     | $d \log d$ , by FFT                             | $d$                                       |

The elementwise product is the cheapest candidate, and the only one exactly undoable at fixed dimension: when every coordinate is  $\pm 1$ ,  $r_i \cdot r_i = 1$ , so multiplying by  $\mathbf{r}$  twice changes nothing. Its price is the one convolution pays as well:  $\mathbf{r} \odot \mathbf{f} = \mathbf{f} \odot \mathbf{r}$ . Session 11's diagnosis, in one line.

## Collecting Is Addition; Marking Is a Permutation

**Collect by adding.** For  $k$  atoms that are orthogonal and of equal length, the sum resembles each one equally:

$$\cos(\mathbf{a}_1 + \cdots + \mathbf{a}_k, \mathbf{a}_j) = \frac{\mathbf{a}_j \cdot \mathbf{a}_j}{\sqrt{k} \|\mathbf{a}_j\|^2} = \frac{1}{\sqrt{k}}, \quad \cos(\mathbf{a}_1 + \cdots + \mathbf{a}_k, \mathbf{b}) = 0 \text{ if } \mathbf{b} \perp \text{ every } \mathbf{a}_i$$

Two atoms: 0.71 each. Three: 0.58. [Membership becomes a threshold on a cosine](#), and since addition is commutative and associative, the result is a set.

**Mark with a permutation.** A permutation matrix  $\Pi$  only reorders coordinates. It is undoable,  $\Pi^{-1} = \Pi^\top$ ; it preserves every dot product,  $\Pi\mathbf{x} \cdot \Pi\mathbf{y} = \mathbf{x} \cdot \mathbf{y}$ ; and for a random  $\mathbf{x}$ ,  $\Pi\mathbf{x}$  is nearly orthogonal to  $\mathbf{x}$ . The standard choice is the cyclic shift:

$$\rho(x_1, x_2, \dots, x_d) = (x_d, x_1, \dots, x_{d-1})$$

So  $\rho(\mathbf{x})$  is a *new* item that can be turned back into  $\mathbf{x}$  on demand, which is what a position tag needs. And because it only reorders, it commutes with every coordinate-wise operation:

$$\rho(\mathbf{x} \odot \mathbf{y}) = \rho(\mathbf{x}) \odot \rho(\mathbf{y}).$$

# MAP, Spelled Out

## MAP: Multiply, Add, Permute (Gayler 1998)

|                        |                                                                                                                                            |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <b>vectors</b>         | $\mathbf{x} \in \{-1, +1\}^d$ ; each coordinate of an atom is $\pm 1$ with probability $\frac{1}{2}$ , independently                       |
| <b>Multiply</b> (bind) | $\mathbf{x} \otimes \mathbf{y} = \mathbf{x} \odot \mathbf{y}$ , that is, $(\mathbf{x} \otimes \mathbf{y})_i = x_i y_i$                     |
| <b>Add</b> (bundle)    | $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = \text{sign}(\mathbf{x} + \mathbf{y} + \mathbf{z})$ , a majority vote in every coordinate |
| <b>Permute</b>         | $\rho(\mathbf{x}) = (x_d, x_1, \dots, x_{d-1})$ , to mark a position or protect a vector                                                   |
| <b>compare</b>         | $\cos(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} / d = 1 - 2h/d$ , with $h$ the number of differing coordinates                 |

**Ties.** With an even number of terms a coordinate can sum to 0; the usual rule breaks the tie at random. **The price of the sign.** Three atoms bundled with the sign score 0.50 each, not the 0.58 of the plain sum. In return the bundle is a  $\pm 1$  vector again, interchangeable with an atom.

**Variants.** **MAP-B**, as above, is this deck's; **MAP-I** keeps the integer sums with no sign; **MAP-C** uses reals in  $[-1, 1]$  and clips the sums. (Names from Schlegel, Neubert & Protzel 2022.)

Every operation is coordinate-wise or a reindexing:  $d$  steps, no FFT, nothing learned.

# MAP's Laws, and What Each One Buys

| Law                               | In symbols                                                                                                     | What it buys                        |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------|-------------------------------------|
| self-inverse                      | $\mathbf{x} \otimes \mathbf{x} = \mathbf{1}$                                                                   | unbinding is binding again          |
| commutative, associative          | $\mathbf{x} \otimes \mathbf{y} = \mathbf{y} \otimes \mathbf{x}$                                                | no brackets – and no argument order |
| distributes over $+$              | $\mathbf{r} \otimes (\mathbf{x} + \mathbf{y}) = \mathbf{r} \otimes \mathbf{x} + \mathbf{r} \otimes \mathbf{y}$ | unbind a record term by term        |
| passes through the sign           | $\mathbf{r} \otimes \text{sign}(\mathbf{s}) = \text{sign}(\mathbf{r} \otimes \mathbf{s})$                      | the same, after bundling            |
| a key keeps similarity            | $(\mathbf{r} \otimes \mathbf{x}) \cdot (\mathbf{r} \otimes \mathbf{y}) = \mathbf{x} \cdot \mathbf{y}$          | similar fillers, similar pairs      |
| a pair resembles neither part     | $(\mathbf{x} \otimes \mathbf{y}) \cdot \mathbf{x} = \sum_i y_i \approx 0$                                      | a pair is a new name                |
| $\rho$ respects $\otimes$ and $+$ | $\rho(\mathbf{x} \otimes \mathbf{y}) = \rho\mathbf{x} \otimes \rho\mathbf{y}$                                  | marking leaves the algebra intact   |

In algebraic terms:  $\{-1, +1\}^d$  under  $\otimes$  is a group in which **every element is its own inverse**, and  $\rho$  is a symmetry of that group. Everything except the sign is linear in each argument; **the sign is the one step that is not**.

Each law is a one-line check, since  $x_i^2 = 1$  and  $\text{sign}(r_i s_i) = r_i \text{sign}(s_i)$ . In the sixth,  $\sum_i y_i \approx \pm\sqrt{d}$  for random  $\mathbf{y}$ , so the cosine is about  $\pm 1/\sqrt{d}$ .

## Worked by Hand: A Codebook, and One Binding

Six atoms at  $d = 8$ , small enough to multiply in your head. Each has four  $+$ 's and four  $-$ 's.

|                                                                               | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |                                              |
|-------------------------------------------------------------------------------|---|---|---|---|---|---|---|---|----------------------------------------------|
| $\mathbf{r}_{\text{pred}}$                                                    | + | + | - | + | - | - | - | + | roles                                        |
| $\mathbf{r}_{\text{ag}}$                                                      | + | - | - | - | + | - | + | + |                                              |
| $\mathbf{r}_{\text{th}}$                                                      | - | + | - | - | - | + | + | + |                                              |
| loves                                                                         | + | - | + | - | + | - | - | + | values                                       |
| john                                                                          | - | - | + | - | - | + | + | + |                                              |
| mary                                                                          | - | + | + | + | - | - | - | + |                                              |
| $\mathbf{r}_{\text{ag}} \otimes \text{john}$                                  | - | + | - | + | - | - | + | + | multiply down each column<br>= john, exactly |
| $(\mathbf{r}_{\text{ag}} \otimes \text{john}) \otimes \mathbf{r}_{\text{ag}}$ | - | - | + | - | - | + | + | + |                                              |

**Against its own factors the pair scores exactly 0:**  $(\mathbf{r}_{\text{ag}} \otimes \text{john}) \cdot \mathbf{r}_{\text{ag}} = \sum_i \text{john}_i = 0$  because john is balanced, and likewise against john, because  $\mathbf{r}_{\text{ag}}$  is. **Binding again undoes it:** each coordinate is multiplied by  $r_i^2 = 1$ .

The roles are mutually orthogonal, and so are the values. Across the two sorts some atoms overlap at  $\pm 0.5$ , and the pair scores  $\pm 0.5$  against the four atoms it was not built from. That is small- $d$  noise: at  $d = 10,000$  the same numbers are about  $\pm 0.01$ .

## Worked by Hand: Bundling a Bag of Words

Bundle the three words of *John loves Mary*: add down each column, then keep the sign.

|              | 1  | 2  | 3 | 4  | 5  | 6  | 7  | 8 |                             |
|--------------|----|----|---|----|----|----|----|---|-----------------------------|
| <b>john</b>  | −  | −  | + | −  | −  | +  | +  | + |                             |
| <b>loves</b> | +  | −  | + | −  | +  | −  | −  | + |                             |
| <b>mary</b>  | −  | +  | + | +  | −  | −  | −  | + |                             |
| <b>sum</b>   | −1 | −1 | 3 | −1 | −1 | −1 | −1 | 3 | add down each column        |
| <b>sign</b>  | −  | −  | + | −  | −  | −  | −  | + | the bundle: a majority vote |

Each word is outvoted in exactly two columns, where the other two words both disagree with it. So each agrees with the bundle in six of eight coordinates:  $\cos = (6 - 2)/8 = 0.50$ . The three roles, which were not bundled, score 0.00.

Now bundle the words of *Mary loves John*. Same three rows, another order: **the same sum, the same vector**. Session 7's bag-of-words failure, in eight coordinates.

Without the sign, the plain sum would score  $8/\sqrt{8 \cdot 24} = 0.58$  against each word: the  $1/\sqrt{k}$  rule of *Collecting Is Addition*, since these three atoms happen to be exactly orthogonal. The sign gives up 0.08 to make the bundle a  $\pm 1$  vector again.

## Worked by Hand: *John Loves Mary* as a Record

Bind each role to its filler (the rows of the codebook slide), bundle the three pairs, and then ask the record a question:

|                                                 | 1 | 2 | 3  | 4  | 5  | 6  | 7 | 8 |            |
|-------------------------------------------------|---|---|----|----|----|----|---|---|------------|
| $\mathbf{r}_{\text{pred}} \otimes \text{loves}$ | + | - | -  | -  | -  | +  | + | + |            |
| $\mathbf{r}_{\text{ag}} \otimes \text{john}$    | - | + | -  | +  | -  | -  | + | + |            |
| $\mathbf{r}_{\text{th}} \otimes \text{mary}$    | + | + | -  | -  | +  | -  | - | + |            |
| sum                                             | 1 | 1 | -3 | -1 | -1 | -1 | 1 | 3 |            |
| $\mathbf{S} = \text{sign}$                      | + | + | -  | -  | -  | -  | + | + | the record |
| $\mathbf{S} \otimes \mathbf{r}_{\text{ag}}$     | + | - | +  | +  | -  | +  | + | + | who loves? |

$\mathbf{S} \otimes \mathbf{r}_{\text{ag}}$  scores 0.50 against john and exactly 0 against the other five atoms, so cleanup returns john. Unbinding with  $\mathbf{r}_{\text{th}}$  returns mary, and with  $\mathbf{r}_{\text{pred}}$  returns loves, each at 0.50 with every other atom at 0.

Why it works: binding passes through the sign, so

$\mathbf{S} \otimes \mathbf{r}_{\text{ag}} = \text{sign}(\text{john} + \mathbf{r}_{\text{ag}} \otimes \mathbf{r}_{\text{pred}} \otimes \text{loves} + \mathbf{r}_{\text{ag}} \otimes \mathbf{r}_{\text{th}} \otimes \text{mary})$  – a bundle that contains john, plus two cross terms that resemble nothing in the codebook.

## Worked by Hand: Permutation, for Protection and for Direction

$\rho$  moves the last coordinate to the front. Two uses, both computed below:

|                                          | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |                                                                                 |
|------------------------------------------|---|---|---|---|---|---|---|---|---------------------------------------------------------------------------------|
| john                                     | - | - | + | - | - | + | + | + | last coordinate moves to the front<br>= 1: both copies gone<br>both copies kept |
| $\rho(\text{john})$                      | + | - | - | + | - | - | + | + |                                                                                 |
| $\text{john} \otimes \text{john}$        | + | + | + | + | + | + | + | + |                                                                                 |
| $\text{john} \otimes \rho(\text{john})$  | - | + | - | - | + | - | + | + |                                                                                 |
| loves                                    | + | - | + | - | + | - | - | + | the arc loves $\rightarrow$ john<br>the arc john $\rightarrow$ loves            |
| $\text{loves} \otimes \rho(\text{john})$ | + | + | - | - | - | + | - | + |                                                                                 |
| $\text{john} \otimes \rho(\text{loves})$ | - | - | - | - | + | + | - | - |                                                                                 |

**Protection.**  $\rho(\text{john})$  scores 0 against john: a new item. Binding john with itself erases both copies, since  $\mathbf{x} \otimes \mathbf{x} = \mathbf{1}$ ; binding it with  $\rho(\text{john})$  keeps them. In Gayler's words (2003), permutation is used "to quote or protect vectors from the other operations."

**Direction.** The two arcs score 0 against each other: as unrelated as two random atoms, although plain binding would have made them identical. To read the arc, unbind loves and shift back:

$$\rho^{-1}((\text{loves} \otimes \rho(\text{john})) \otimes \text{loves}) = \text{john}.$$

# Three Operations, and What Each Preserves

Part III built the three **MAP** operations by hand at  $d = 8$ . From here on  $d = 10,000$ , where random atoms are nearly orthogonal and the rough cosines of the hand examples settle down. Every step is still exact integer arithmetic.

|                | Written                                                               | Result is similar to | Use                    |
|----------------|-----------------------------------------------------------------------|----------------------|------------------------|
| <b>bundle</b>  | $\mathbf{a} \oplus \mathbf{b} = \text{sign}(\mathbf{a} + \mathbf{b})$ | both inputs          | a set, a superposition |
| <b>bind</b>    | $\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \odot \mathbf{b}$         | neither input        | a pair, a role-filler  |
| <b>permute</b> | $\rho(\mathbf{a}) = \text{shift by one}$                              | not the input        | order, position        |

The asymmetry between the first two is the whole design. Bundling *preserves* similarity, so a bundle is recognizably each of its parts. Binding *destroys* it, so a binding is a new name rather than a blend.

Bundling is the addition distributional semantics already had; binding is the operation it never had.

## Bundling: a Set as One Vector

Encode the content words of *Alice bought a book* as one vector, at  $d = 10,000$ :

$$\mathbf{s} = \mathbf{alice} \oplus \mathbf{bought} \oplus \mathbf{book}$$

| $\cos(\mathbf{s}, \cdot)$ | alice | bought | book | mary | sold |
|---------------------------|-------|--------|------|------|------|
|                           | 0.50  | 0.49   | 0.51 | 0.01 | 0.02 |

Membership is a threshold test with an enormous margin, and the vector has the same dimension as each of its parts. You can ask “was book in this sentence?” in one dot product.

And you can ask nothing else. The bundle is commutative and associative, so it records the set  $\{\text{alice}, \text{bought}, \text{book}\}$  – no order, no counts, no argument structure. This is precisely the bag-of-words representation from Session 7, rebuilt from the other end, and it fails in precisely the same way.

So bundling alone is not progress. The progress is that we now have a second operation to pair it with.

## Binding: a Pair as One Vector

For bipolar vectors, binding is elementwise multiplication, and it has two properties that make it usable:

$$\mathbf{a} \otimes \mathbf{b} \perp \mathbf{a}, \mathbf{b} \quad \text{and} \quad (\mathbf{a} \otimes \mathbf{b}) \otimes \mathbf{a} = \mathbf{b}$$

The second holds *exactly*, since  $\mathbf{a} \odot \mathbf{a} = \mathbf{1}$ . Binding is its own inverse, so there is no approximation and no need for the unitary trick that HRR requires.

**But read the operation again.**  $\mathbf{a} \odot \mathbf{b} = \mathbf{b} \odot \mathbf{a}$ . Elementwise multiplication is **commutative**, exactly like circular convolution. This is the loss from Session 11, and it has not gone away by changing the binding operation.

| Binding operation                               | Commutative? | Cost  |
|-------------------------------------------------|--------------|-------|
| outer product $\mathbf{r}\mathbf{f}^\top$ (TPR) | no           | $n^2$ |
| circular convolution (HRR)                      | yes          | $n$   |
| elementwise product (MAP)                       | yes          | $n$   |

Every binding operation that stays in dimension  $n$  in common use is commutative. The next slide is how the order gets back in.

# Where the Order Actually Comes From

Two devices, and neither is the binding operation.

**1. Stipulate the roles.** The device from last session, and it transfers unchanged to MAP:

$$\text{loves}(j, m) \mapsto (\mathbf{r}_{\text{agent}} \otimes \mathbf{john}) \oplus (\mathbf{r}_{\text{theme}} \otimes \mathbf{mary})$$

The order is in the vocabulary, not in the algebra – and the cost, as we said, is that the role inventory has to be fixed in advance by the designer.

**2. Permute.** Let  $\rho$  be a fixed coordinate shuffle – a cyclic shift will do. moving  $\rho$  from one factor of a binding to the other changes the result,  $\mathbf{x} \otimes \rho(\mathbf{y}) \neq \rho(\mathbf{x}) \otimes \mathbf{y}$ , and  $\rho^i$  marks position  $i$ . The next slide works this one out.

Both devices do the same job: they put the asymmetry somewhere the commutative binding cannot reach. Roles put it in the vocabulary; permutation puts it in the operation.

## Sequences, by Permutation

To encode the string  $a\ b\ c$  as one vector, shift each atom by its position and bundle:

$$\mathbf{s} = \rho^0(\mathbf{a}) \oplus \rho^1(\mathbf{b}) \oplus \rho^2(\mathbf{c})$$

To read position  $i$ , shift back by  $i$  and clean up. Measured at  $d = 10,000$ :

| query                   | cos to <b>a</b> | cos to <b>b</b> | cos to <b>c</b> |
|-------------------------|-----------------|-----------------|-----------------|
| $\rho^0(\mathbf{s})$    | +0.504          | +0.005          | -0.003          |
| $\rho^{-1}(\mathbf{s})$ | +0.000          | +0.490          | +0.007          |
| $\rho^{-2}(\mathbf{s})$ | +0.012          | +0.002          | +0.501          |

Each position returns its own item at about 0.5 and everything else at the noise floor. **This is an  $n$ -gram representation that does not grow with  $n$** , and unlike a bag of words it distinguishes  $abc$  from  $cba$ .

Repetition is free:  $a$  occurring at positions 0 and 3 contributes  $\rho^0(\mathbf{a})$  and  $\rho^3(\mathbf{a})$ , which are near-orthogonal, so the two occurrences do not interfere. **A bundle could not have done that – it would have counted  $a$  once.**

# In-Class Exercise 1

Work with bipolar vectors and  $\otimes = \odot$  throughout.

- 1 Show that  $\mathbf{a} \otimes \mathbf{a} = \mathbf{1}$  for every bipolar  $\mathbf{a}$ , and say why that makes unbinding exact.
- 2 You are told  $\mathbf{x} = \mathbf{r}_{\text{agent}} \otimes \mathbf{john}$ . Someone hands you  $\mathbf{x}$  and  $\mathbf{john}$  and asks which role was used. How do you find out, and what do you need to have kept?
- 3 Bundle is commutative and bind is commutative. Explain, in one sentence each, why *John loves Mary* and *Mary loves John* nevertheless receive different vectors under the role-stipulation scheme.
- 4 ★ A colleague proposes dropping roles and encoding the sentence as  $\mathbf{john} \otimes \mathbf{loves} \otimes \mathbf{mary}$ . What goes wrong, and what exactly can still be recovered from that vector?

Six minutes. Item 4 is the starred one.

## In-Class Exercise 1: Answers

1. Each coordinate is  $\pm 1$ , so  $a_i^2 = 1$  for every  $i$ . Hence  $(\mathbf{a} \otimes \mathbf{b}) \otimes \mathbf{a} = \mathbf{b} \odot (\mathbf{a} \odot \mathbf{a}) = \mathbf{b}$ , **exactly, at every dimension** – no noise term at all for a single bound pair.
2. Compute  $\mathbf{x} \otimes \mathbf{john}$ , which is  $\mathbf{r}_{\text{agent}}$  exactly, then compare it against your stored role atoms. **You need the codebook.** Without a list of the atoms you minted, the result is an uninterpretable  $\pm 1$  vector – which is the point of Part V.
3. In *John loves Mary* the pair  $\mathbf{r}_{\text{agent}} \otimes \mathbf{john}$  appears; in the reverse sentence  $\mathbf{r}_{\text{agent}} \otimes \mathbf{mary}$  appears instead. **Commutativity says the two factors inside one binding may be swapped; it does not say two different bindings may be swapped.** The asymmetry was moved into the role atoms, which differ.
4. ★ All three are bound into one atom-like vector with no slots. You can still ask *whether* a given atom participated, by unbinding the other two – so the vector records the set  $\{\mathbf{john}, \mathbf{loves}, \mathbf{mary}\}$  and nothing more. **It is a bag of words with extra steps**, and it is the same failure as Session 7, now at the level of a single binding.

# Unbinding Returns Noise, Not an Answer

Build a record with three role-filler pairs, then unbind one role:

$$\mathbf{R} = (\mathbf{r}_1 \otimes \mathbf{f}_1) \oplus (\mathbf{r}_2 \otimes \mathbf{f}_2) \oplus (\mathbf{r}_3 \otimes \mathbf{f}_3), \quad \mathbf{R} \otimes \mathbf{r}_2 = \mathbf{f}_2 + \underbrace{\text{two cross terms}}_{\text{noise}}$$

The cross terms are bindings of unrelated atoms, so they are orthogonal to everything you care about – but they are *there*, and the result is not  $\mathbf{f}_2$ .

Note where this noise comes from, because it is not where it came from last session. MAP's binding is *exactly* invertible:  $\mathbf{a} \otimes \mathbf{a} = \mathbf{1}$ , with no approximation at all. Every bit of the noise below is contributed by the superposition, not by the binding – whereas HRR's convolution is itself only approximately invertible and adds noise of its own on top.

|                        | $\cos(\mathbf{R} \otimes \mathbf{r}_2, \mathbf{f}_2)$ | cos to any other atom |
|------------------------|-------------------------------------------------------|-----------------------|
| measured, $d = 10,000$ | 0.50                                                  | $< 0.01$              |

A cosine of 0.50 is not a filler. It is a vector that *resembles* a filler. If you feed it into the next operation, the noise compounds, and a few steps later nothing is recoverable.

So a VSA needs one more component, and it is not an implementation detail.

# Cleanup Memory: Where the Symbol Comes Back

## Definition

A **cleanup memory** (or item memory) is the stored list of every atom you minted. Given a noisy vector  $\mathbf{q}$ , it returns the single stored atom nearest to  $\mathbf{q}$ :

$$\text{cl}(\mathbf{q}) = \arg \max_{\mathbf{v} \in \text{codebook}} \cos(\mathbf{q}, \mathbf{v})$$

Applied to the previous slide:  $\text{cl}(\mathbf{R} \otimes \mathbf{r}_2) = \mathbf{f}_2$ , **exactly**, with the noise discarded.

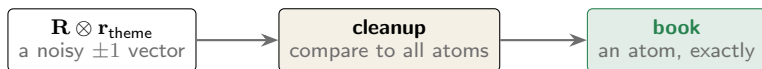
**This is the conceptually important step of the whole architecture.** Everything before cleanup is continuous: vectors, sums, products, graded cosines. Cleanup is a *nearest-neighbor snap to a finite inventory*, and it is the moment a vector becomes a symbol again.

So the answer to “where does the discreteness live?” is not in the vectors and not in the operations. **It lives in the codebook, and in the decision to round to it.** A VSA without cleanup is an analog system that drifts; with cleanup it is a symbol processor.

In neural terms cleanup is an associative memory, and it is usually what the Hopfield-style or attention-style layer is doing in hybrid models.

## Cleanup, Worked

The record from two slides back, with the codebook listed beside it. Query the theme slot:



cos: book 0.50 | alice 0.01 | buy 0.00 | mary -0.01

The input to cleanup is a vector no human could read. The output is a member of a finite, known inventory. **Nothing in between was symbolic; the output is.**

Note what cleanup needs: **the list**. The architecture is not self-contained – it presupposes that you kept a record of every atom you ever minted, and that the thing you are looking for is on it. A VSA cannot recognize a symbol it was never given.

Which is a real constraint for language, where the inventory is open: new names, new words, new senses.

## Capacity: How Much Fits Before Cleanup Fails

Bundle  $k$  role-filler pairs into one vector, unbind one role, and ask whether cleanup returns the right filler. Measured accuracy over 30 trials:

| $d$    | $k = 10$ | $k = 50$ | $k = 100$ | $k = 200$ | $k = 400$ |
|--------|----------|----------|-----------|-----------|-----------|
| 256    | 0.97     | 0.43     | 0.03      | 0.00      | 0.00      |
| 1,000  | 1.00     | 0.90     | 0.60      | 0.13      | 0.00      |
| 10,000 | 1.00     | 1.00     | 1.00      | 1.00      | 0.83      |

Capacity grows roughly linearly in  $d$ , and the failure is a cliff rather than a slope: at  $d = 256$  the system goes from 97% to 3% between  $k = 10$  and  $k = 100$ .

Two consequences for anyone building with this. **First**, you must size  $d$  to the largest structure you intend to hold, because there is no graceful degradation. **Second**, deep nesting is expensive: each level of binding adds noise, so a tree of depth 5 needs far more room than 5 flat pairs.

This is the honest version of “it is robust.” It is robust to coordinate noise and intolerant of overload.

## A Record, and an Event

The record is the basic structure, and it is exactly the role–filler table from Session 10's Part IV, now compressed to one vector.

$$\mathbf{E} = (\mathbf{r}_{\text{pred}} \otimes \mathbf{buy}) \oplus (\mathbf{r}_{\text{agent}} \otimes \mathbf{alice}) \oplus (\mathbf{r}_{\text{theme}} \otimes \mathbf{book}) \oplus (\mathbf{r}_{\text{source}} \otimes \mathbf{mary})$$

That is *Alice bought a book from Mary* as a single vector of  $d$  numbers, for any  $d$  you choose, with four slots you can interrogate independently:

| Query                     | Compute                                                   | Returns                      |
|---------------------------|-----------------------------------------------------------|------------------------------|
| who was the agent?        | $\text{cl}(\mathbf{E} \otimes \mathbf{r}_{\text{agent}})$ | alice                        |
| what was the predicate?   | $\text{cl}(\mathbf{E} \otimes \mathbf{r}_{\text{pred}})$  | buy                          |
| what role does mary fill? | $\text{cl}(\mathbf{E} \otimes \mathbf{mary})$             | $\mathbf{r}_{\text{source}}$ |

Note the third row. The scheme is symmetric in use: you can unbind by role to get the filler, or unbind by filler to get the role. That is a property the role–filler table had too, by reading a row instead of a column.

Adjuncts attach the same way, as further conjuncts, which is the event-semantics picture from Session 10:  $\oplus$  is conjunction over an event variable that never has to be named.

## John loves Mary, End to End

The challenge from last session, now answerable. Encode both sentences as records:

$$\mathbf{S}_1 = (\mathbf{r}_{\text{pred}} \otimes \text{loves}) \oplus (\mathbf{r}_{\text{ag}} \otimes \text{john}) \oplus (\mathbf{r}_{\text{th}} \otimes \text{mary})$$

$$\mathbf{S}_2 = (\mathbf{r}_{\text{pred}} \otimes \text{loves}) \oplus (\mathbf{r}_{\text{ag}} \otimes \text{mary}) \oplus (\mathbf{r}_{\text{th}} \otimes \text{john})$$

|                                        |                    | from $\mathbf{S}_1$ |        | from $\mathbf{S}_2$ |        |
|----------------------------------------|--------------------|---------------------|--------|---------------------|--------|
| query                                  |                    | john                | mary   | john                | mary   |
| $\cdot \otimes \mathbf{r}_{\text{ag}}$ | <i>who loves?</i>  | +0.503              | -0.013 | -0.001              | +0.491 |
| $\cdot \otimes \mathbf{r}_{\text{th}}$ | <i>loves whom?</i> | +0.003              | +0.494 | +0.507              | -0.010 |

|                                    |                                     |       |                                     |
|------------------------------------|-------------------------------------|-------|-------------------------------------|
| $\cos(\mathbf{S}_1, \mathbf{S}_2)$ | as records                          | 0.496 | – related but <i>distinct</i>       |
|                                    | cos of the two <b>bags of words</b> | 1.000 | – identical, as Session 7 predicted |

Fodor and Pylyshyn's challenge, met: the two sentences are told apart, each argument slot is independently queryable, and the operations did not change. The 0.496 is right too – these sentences *should* be similar, sharing a predicate and a cast.

## Nesting, and Why It Is Not Free

Fillers may themselves be records, so you get trees:

$$\mathbf{S} = (\mathbf{r}_{\text{pred}} \otimes \text{say}) \oplus (\mathbf{r}_{\text{agent}} \otimes \text{john}) \oplus (\mathbf{r}_{\text{comp}} \otimes \underbrace{\mathbf{E}}_{\text{Alice bought a book}})$$

*John said that Alice bought a book from Mary*, still one vector of dimension  $d$ . The embedded clause is a filler like any other, which is the systematicity Fodor and Pylyshyn asked for: **the representation is recursive and the operations do not change with depth.**

**The cost.** To reach alicia you unbind twice, and each unbinding adds its own cross-term noise. Signal falls by a factor of roughly  $1/\sqrt{k}$  per level with  $k$  conjuncts – for  $k = 3$ , by half – so depth multiplies:

| depth                                           | 1    | 2    | 3    | 4    |
|-------------------------------------------------|------|------|------|------|
| cos to the target filler, 3 conjuncts per level | 0.50 | 0.25 | 0.12 | 0.06 |

By depth 4 you are at 0.06 against a noise floor of 0.01 at  $d = 10,000$  – still readable, but the margin is gone two levels later. Cleanup at every level is what keeps this usable: snap to the codebook before descending further.

# The Structure Zoo, in One Table

Everything standard is built from the same three operations. Nothing below needs a new primitive.

| Structure     | Encoding                                                | Query by                       |
|---------------|---------------------------------------------------------|--------------------------------|
| set           | $\mathbf{a} \oplus \mathbf{b} \oplus \mathbf{c}$        | dot product, threshold         |
| pair          | $\mathbf{a} \otimes \mathbf{b}$                         | unbind with either half        |
| record        | $\bigoplus_i \mathbf{r}_i \otimes \mathbf{f}_i$         | unbind with a role             |
| sequence      | $\bigoplus_i \rho^i(\mathbf{a}_i)$                      | unshift by position            |
| tree          | records whose fillers are records                       | unbind, clean up, descend      |
| labeled graph | $\bigoplus_{(u,v)} \mathbf{u} \otimes \rho(\mathbf{v})$ | $\rho$ makes the edge directed |

Look at the last row. An edge  $u \rightarrow v$  is encoded as  $\mathbf{u} \otimes \rho(\mathbf{v})$ , which differs from  $\mathbf{v} \otimes \rho(\mathbf{u})$ . That is how a dependency arc gets its arrowhead – the permutation, not the binding, carries the direction.

So a whole dependency parse is one vector: bundle one term per arc, each arc a bound pair with the dependent permuted. Querying “what is the nsubj of loves?” is two operations and a cleanup.

## A Lexical Entry Is a Record

Nothing in the machinery cares whether the roles are thematic. A structured lexical entry encodes exactly the same way – here a qualia structure for novel:

$$\mathbf{novel} = (\mathbf{r}_{\text{formal}} \otimes \mathbf{book}) \oplus (\mathbf{r}_{\text{const}} \otimes \mathbf{narrative}) \oplus (\mathbf{r}_{\text{telic}} \otimes \mathbf{read}) \oplus (\mathbf{r}_{\text{agentive}} \otimes \mathbf{write})$$

Four roles, four fillers, one vector.  $\text{cl}(\mathbf{novel} \otimes \mathbf{r}_{\text{telic}})$  returns read.

Which makes a concrete point about what this architecture is for. A VSA is a representation for structures you have already analyzed. It will faithfully store a qualia structure, a frame, an AMR, a feature bundle – and it will not tell you what the roles should be or which filler belongs in one.

That division of labor is worth naming. The theory supplies the inventory and the slots; the algebra supplies storage, composition, and retrieval at fixed dimension. The same split holds for the lexicon: distributional methods learn what novel is like, a structured theory says what parts it has, and a VSA holds the result in a form you can compute with.

Compare the matrix encoding of the same information in Session 10: the roles were columns of a table. Here they are atoms, and the table has been folded into one vector.

## In-Class Exercise 2: Structures

- 1 Encode the dependency arc  $\text{loves} \xrightarrow{\text{nsubj}} \text{John}$  two ways: as a labeled binding  $\mathbf{r}_{\text{nsubj}} \otimes \text{loves} \otimes \text{john}$ , and as  $\text{loves} \otimes \rho(\text{john})$ . What can you query from each?
- 2 A record holds agent and theme. You want to add a second theme – a coordination, *Alice bought a book and a pen*. What goes wrong, and what is the standard fix?
- 3 Why is a sequence encoded with  $\rho$  able to handle a repeated letter, while a bundle cannot handle a repeated word?
- 4 ★ The cleanup memory must contain every atom. For a language model that is an open vocabulary: new names, new words, new senses. Sketch what breaks, and say whether you think it is a fixable engineering problem or a claim about what this architecture can be.

Seven minutes. Item 4 is the starred one.

## In-Class Exercise 2: Answers

1. The labeled binding lets you recover any one of the three given the other two, but it records an *unordered* triple – the arc direction is carried only by the label  $\mathbf{r}_{\text{nsbj}}$ . *The permuted form loves  $\otimes \rho(\text{john})$  carries direction in the operation* and differs from  $\text{john} \otimes \rho(\text{loves})$ , but it has no slot for the label. Real systems use both:  $\mathbf{r}_{\text{nsbj}} \otimes \text{loves} \otimes \rho(\text{john})$ .
2.  $\mathbf{r}_{\text{th}} \otimes \text{book} \oplus \mathbf{r}_{\text{th}} \otimes \text{pen}$  unbinds to  $\text{book} \oplus \text{pen}$ , which cleanup will resolve to *one* of them arbitrarily. *The fix is to bundle first and bind once*,  $\mathbf{r}_{\text{th}} \otimes (\text{book} \oplus \text{pen})$ , and to have cleanup return a *set* by thresholding rather than taking the argmax.
3.  $\rho^0(\mathbf{a})$  and  $\rho^3(\mathbf{a})$  are near-orthogonal, so two occurrences of  $\mathbf{a}$  occupy independent directions and do not interfere. *A bundle adds  $\mathbf{a}$  to itself, and  $\text{sign}(2\mathbf{a}) = \mathbf{a}$  – the second copy leaves no trace.* Position is what makes repetition representable.
4. ★ An unseen atom is not in the codebook, so cleanup returns the nearest *stored* atom: a confident wrong answer, with no signal that anything is amiss. You can grow the codebook, but every addition raises the chance of a collision and lowers capacity. *Whether that is engineering or architecture is a fair thing to argue about; my own view is that it marks the boundary between a system that stores structure and one that acquires it.*

## Analogy With No Training At All

Kanerva's example, and the thing to watch is that there is no learning anywhere in it. Two records, built as on the previous slides:

$$\mathbf{USA} = (\mathbf{r}_{\text{cap}} \otimes \mathbf{wdc}) \oplus (\mathbf{r}_{\text{cur}} \otimes \mathbf{dollar}) \oplus (\mathbf{r}_{\text{ctry}} \otimes \mathbf{usa})$$

$$\mathbf{MEX} = (\mathbf{r}_{\text{cap}} \otimes \mathbf{mxo}) \oplus (\mathbf{r}_{\text{cur}} \otimes \mathbf{peso}) \oplus (\mathbf{r}_{\text{ctry}} \otimes \mathbf{mex})$$

Now bind the two records together and see what the result is made of:

$$\mathbf{F} = \mathbf{USA} \otimes \mathbf{MEX} \approx (\mathbf{wdc} \otimes \mathbf{mxo}) \oplus (\mathbf{dollar} \otimes \mathbf{peso}) \oplus (\mathbf{usa} \otimes \mathbf{mex}) \oplus \text{noise}$$

The matching roles cancel, because  $\mathbf{r} \otimes \mathbf{r} = \mathbf{1}$ . What survives is a bundle of **corresponding fillers paired with each other** – a mapping between the two countries, assembled by one multiplication.

So **F** is a **reusable analogy**. Bind it with anything from one country and it hands back the counterpart in the other.

# What Is the Dollar of Mexico?

$$\mathbf{F} \otimes \text{dollar} \approx \text{peso} \oplus \text{noise}$$

Measured at  $d = 10,000$  with nine random atoms. Cosine of the answer against every atom in the codebook:

| peso          | currency | usa    | dollar | country | mex    | capital | mxc    | wdc    |
|---------------|----------|--------|--------|---------|--------|---------|--------|--------|
| <b>+0.242</b> | +0.019   | +0.001 | -0.001 | -0.003  | -0.003 | -0.007  | -0.014 | -0.014 |

The gap is a factor of twelve. Cleanup returns peso with no ambiguity whatsoever.

## What just happened

No gradient descent, no corpus, no training objective, no learned parameters. Nine random vectors, three operations, and one algebraic expression. **The analogy was computed, not learned.**

Contrast the word2vec analogy result, **king** - **man** + **woman**  $\approx$  **queen**, which requires a trained space and works unevenly. Here it is exact algebra, and it works because the structure was *put in* rather than hoped for.

## In-Class Exercise 3

- 1 Using the same  $\mathbf{F}$ , what does  $\mathbf{F} \otimes \mathbf{mxc}$  return? Predict before computing.
- 2 Show algebraically why the role atoms vanish from  $\mathbf{USA} \otimes \mathbf{MEX}$ , and identify how many noise terms survive for records of 3 pairs each.
- 3 The analogy vector  $\mathbf{F}$  is symmetric:  $\mathbf{USA} \otimes \mathbf{MEX} = \mathbf{MEX} \otimes \mathbf{USA}$ . Is that a problem for using it as a mapping? Why or why not?
- 4 ★ Build  $\mathbf{F}$  from two records with *different* role sets – say the second record has no  $\mathbf{r}_{\text{cur}}$  slot. What happens to the query  $\mathbf{F} \otimes \mathbf{dollar}$ , and what does that tell you about how much this method assumes about the alignment of the two structures?

Seven minutes. Item 4 is the starred one, and it is where the method's real limit is.

## In-Class Exercise 3: Answers

1. **wdc** – Washington. The term  $\mathbf{wdc} \otimes \mathbf{mxc}$  is present in  $\mathbf{F}$ , and binding with **mxc** cancels it. *The same  $\mathbf{F}$  answers every corresponding-entity question in both directions.*
2. Expanding the product of two 3-term bundles gives 9 terms. The 3 with matching roles contribute  $\mathbf{r} \otimes \mathbf{r} \otimes \mathbf{f} \otimes \mathbf{f}' = \mathbf{f} \otimes \mathbf{f}'$ ; the other 6 are bindings of four unrelated atoms, hence orthogonal to everything in the codebook. *Signal 3, noise 6 – which is why the answer came back at 0.24 rather than 0.50.*
3. Not a problem, and arguably a feature: the mapping is a correspondence, not a function with a direction, and item 1 used it both ways. *But note what this means –  $\mathbf{F}$  cannot encode an asymmetric analogy.* For *dog is to puppy as cat is to kitten* the adult-to-young direction has to be carried by the roles, not by  $\mathbf{F}$ .
4. ★ The  $\mathbf{r}_{\text{cur}}$  term has no partner to cancel against, so **dollar** contributes only noise and the query returns nothing above the floor. *The method requires the two structures to be aligned role-for-role in advance.* It computes analogies between structures that already match; it does not *find* the alignment, which is the hard part of analogy.

## Every Type in One Space

In Sessions 6 and 11b a type fixed the shape of its representation: order = the number of  $e$ -slots, and application = contraction. A VSA sends every type to the same place.

| What is represented                                   | tensor framework                      | VSA         |
|-------------------------------------------------------|---------------------------------------|-------------|
| an individual, $e$                                    | a vector, order 1                     | $d$ numbers |
| a predicate, $\langle e, t \rangle$                   | a vector, order 1                     | $d$ numbers |
| a relation, $\langle e, \langle e, t \rangle \rangle$ | a matrix, order 2                     | $d$ numbers |
| a role-filler pair                                    | a matrix $\mathbf{rf}^\top$ , order 2 | $d$ numbers |
| a clause inside a clause                              | order 3                               | $d$ numbers |

Session 6 called this the instructive failure of the VSA row: “you cannot read a predicate’s arity off its representation.” **The vector does not carry its type.**

Logic has a name for the difference. A **Church-style** system builds the type into each term: a matrix cannot stand where a vector is expected. A **Curry-style** system has untyped terms and assigns types by rules about how terms are built and used. **Tensors are typed the first way; a VSA can only be typed the second.**

## Roles and Values: The Codebook Is the Signature

Two base types do almost all the work: **values**, the fillers (john, book, and predicate names like loves), and **roles** (agent, theme, pred). Both are random  $\pm 1$  draws, so nothing in the vectors tells them apart. What does is the list each was filed in:

$$\Sigma = \{ \text{john: Val, mary: Val, loves: Val, r}_{\text{ag}}\text{: Role, r}_{\text{th}}\text{: Role, r}_{\text{pred}}\text{: Role} \}$$

In type theory a **signature** lists the constants and the type of each. The cleanup memory, partitioned by sort, is exactly that.

So cleanup should be typed as well. A question that asks for a value searches only the values:

$$\text{cl}_T(\mathbf{q}) = \arg \max_{\mathbf{v}: T \text{ in } \Sigma} \cos(\mathbf{q}, \mathbf{v})$$

*Who loves?* calls  $\text{cl}_{\text{Val}}$ ; *which role does mary fill?* calls  $\text{cl}_{\text{Role}}$ . Two gains: a role can never come back as the answer to a question about values, and fewer candidates means fewer near misses, which is more capacity at the same  $d$ .

## Typing Rules for the Operations

A **record type** lists labels and the type stored under each, such as {agent: Val, theme: Val}; the role atoms serve as the labels. Each operation then gets a rule:

$$\begin{array}{c} \frac{\mathbf{r} : \text{Role} \quad \mathbf{v} : T}{\mathbf{r} \otimes \mathbf{v} : \{\mathbf{r} : T\}} \text{ BIND} \qquad \frac{\mathbf{x} : \{\dots, \mathbf{r} : T, \dots\}}{\text{cl}_T(\mathbf{x} \otimes \mathbf{r}) : T} \text{ SELECT} \\[2ex] \frac{\mathbf{x} : \{\vec{\ell} : \vec{T}\} \quad \mathbf{y} : \{\vec{m} : \vec{U}\} \quad \vec{\ell} \cap \vec{m} = \emptyset}{\mathbf{x} \oplus \mathbf{y} : \{\vec{\ell} : \vec{T}, \vec{m} : \vec{U}\}} \text{ RECORD} \qquad \frac{\mathbf{v}_1 : T \quad \dots \quad \mathbf{v}_k : T}{\mathbf{v}_1 \oplus \dots \oplus \mathbf{v}_k : \text{Set}(T)} \text{ SET} \end{array}$$

One operation,  $\oplus$ , has two rules: on fields it builds a record, on values of one type it builds a set. The types of its arguments decide which rule applies.

**The side condition earns its place.** *Alice bought a book and a pen* puts two values under one label, violating  $\vec{\ell} \cap \vec{m} = \emptyset$ , and cleanup returns one of them arbitrarily: Exercise 2, item 2. The fix given there is the typed one. Build a set, then bind it once:  $\mathbf{r}_{\text{th}} \otimes (\mathbf{book} \oplus \mathbf{pen}) : \{\text{theme} : \text{Set}(\text{Val})\}$ . The field's type then tells cleanup to threshold instead of taking the argmax.

## Two Readings of the Same Operations

The rules can be read as records in a programming language or, since a record like  $\mathbf{E}$  describes an event, in the neo-Davidsonian logic of Session 10, with events of type  $v$ . Read that way, a role is a function from an event to its participant.

| VSA                                                                 | as a record                                  | as event semantics                                                                    |
|---------------------------------------------------------------------|----------------------------------------------|---------------------------------------------------------------------------------------|
| <b>john</b>                                                         | a constant of type $\text{Val}$              | an individual, type $e$                                                               |
| $\mathbf{r}_{\text{ag}}$                                            | a label                                      | a role function, agent : $\langle v, e \rangle$                                       |
| $\mathbf{r}_{\text{ag}} \otimes \mathbf{john}$                      | a one-field record                           | the condition $\text{agent}(\varepsilon) = \text{john}$ , type $\langle v, t \rangle$ |
| $\mathbf{x} \oplus \mathbf{y}$ , on fields                          | joining two records                          | conjunction, $\lambda \varepsilon. P(\varepsilon) \wedge Q(\varepsilon)$              |
| $\text{cl}_{\text{Val}}(\mathbf{E} \otimes \mathbf{r}_{\text{ag}})$ | selecting a field, $\mathbf{E}.\text{agent}$ | <i>the</i> agent of the event $\mathbf{E}$ describes                                  |
| $\mathbf{a} \oplus \mathbf{b} \oplus \mathbf{c}$ , on values        | a set                                        | a plurality, or a predicate $\langle e, t \rangle$                                    |

The deck's earlier remark, made exact:  $\oplus$  is conjunction over an event variable that never has to be named. And cleanup plays the definite article. It returns *the* value and presupposes there is exactly one, which is why a second theme broke it on the last slide.

## A Role Is a Function, Stored as Its Diagonal

Fix a role, and binding with it is a linear map of the whole space:

$$\mathbf{x} \longmapsto \mathbf{r} \otimes \mathbf{x} = \text{diag}(\mathbf{r}) \mathbf{x}, \quad \text{diag}(\mathbf{r})^2 = I$$

So a role *is* an order-2 object in Session 11b's sense: a  $d \times d$  matrix, stored as its diagonal. Since its square is  $I$ , **the same matrix builds the field and reads it back**; only the typing rules keep the two uses apart.

HRR stores roles the same way in another basis: binding with  $\mathbf{u}$  is the circulant  $C(\mathbf{u})$ , and  $\mathcal{F} C(\mathbf{u}) \mathcal{F}^{-1} = \text{diag}(\mathcal{F}\mathbf{u})$  (Session 11f). So the VSA families are one algebra:

| family | coordinates                                                | binding                                        |
|--------|------------------------------------------------------------|------------------------------------------------|
| FHRR   | unit complex numbers $e^{i\theta}$                         | multiply coordinate-wise: add the phases       |
| HRR    | reals, in the position basis                               | convolve: multiply in the Fourier basis        |
| MAP-B  | $\pm 1$ , that is $e^{i0}$ and $e^{i\pi}$                  | multiply coordinate-wise: FHRR with two phases |
| BSC    | bits, with $0 \leftrightarrow +1$ , $1 \leftrightarrow -1$ | XOR, which is MAP-B's product relabeled        |

Bundling matches too: FHRR adds and rescales each coordinate back to the unit circle, and on the real line that rescaling is the sign.

# Typing the Structure Zoo

| Structure     | Encoding                                         | Type                                                            | A query returns                            |
|---------------|--------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------|
| set           | $\mathbf{a} \oplus \mathbf{b} \oplus \mathbf{c}$ | $\text{Set}(\text{Val})$                                        | yes/no: a predicate $\langle e, t \rangle$ |
| pair          | $\mathbf{a} \otimes \mathbf{b}$                  | an <i>unordered</i> pair                                        | the other value                            |
| record        | $\bigoplus_i \mathbf{r}_i \otimes \mathbf{f}_i$  | $\{\mathbf{r}_1: \text{Val}, \dots, \mathbf{r}_k: \text{Val}\}$ | a value, by label                          |
| sequence      | $\bigoplus_i \rho^i(\mathbf{a}_i)$               | a tuple, $\{0: \text{Val}, 1: \text{Val}, \dots\}$              | a value, by position                       |
| directed edge | $\mathbf{u} \otimes \rho(\mathbf{v})$            | $\text{Val} \times \text{Val}$ , ordered                        | a value                                    |
| tree          | records within records                           | a nested record type                                            | a value, by path                           |

Roles supply named labels;  $\rho^i$  supplies numbered ones. A tuple is a record whose labels are positions, which is the standard encoding in type theory.

The unlabeled product is the odd one out. Since  $\mathbf{a} \otimes \mathbf{b} = \mathbf{b} \otimes \mathbf{a}$ , it is an unordered pair, Session 11e's  $\text{Sym}^2$ ; and  $\mathbf{john} \otimes \mathbf{loves} \otimes \mathbf{mary}$  is an unordered triple, Exercise 1's “bag of words with extra steps,” now with a type. Worse,  $\mathbf{a} \otimes \mathbf{a} = \mathbf{1}$ : a pair of two copies vanishes unless one copy is protected,  $\mathbf{a} \otimes \rho(\mathbf{a})$ .

## Paths, and What Permutation Protects

A nested record has typed **paths**. In *John said that Mary saw Bill* the theme is a record, and mary sits at the path theme.agent:

$S : \{\text{pred: Val, agent: Val, theme: \{\text{pred: Val, agent: Val, theme: Val}\}}\}$

MAP follows a path by unbinding its roles in turn, and  $\otimes$  is commutative and self-inverse. So **a path is only a set of roles, with repeats canceled**: theme.agent and agent.theme are one key, and theme.theme is no key at all.

| query on S                | should give               | plain MAP | with $\rho(\mathbf{E})$ |
|---------------------------|---------------------------|-----------|-------------------------|
| theme.agent, against mary | signal                    | 0.25      | 0.25                    |
| agent.theme, against mary | noise (the agent is john) | 0.25      | 0.00                    |
| S itself, against bill    | noise (two levels down)   | 0.25      | 0.00                    |
| theme.theme, against bill | signal                    | 0.25      | 0.25                    |

Mean of 100 draws at  $d = 10,000$ . The fix is Gayler's "quote or protect." With each embedded record stored as  $\rho(\mathbf{E})$ , the path  $\ell_1.\ell_2$  is filed under  $\mathbf{r}_{\ell_1} \otimes \rho(\mathbf{r}_{\ell_2})$ , which is **ordered**, and an ill-typed path returns noise instead of a confident wrong answer.

## The Analogy, Typed

The two country records share one record type,  $\mathbf{USA}, \mathbf{MEX} : \{\text{cap: Val}, \text{cur: Val}, \text{ctry: Val}\}$ . Binding them cancels every shared label and pairs the values that sat under it:

$$\mathbf{F} = \mathbf{USA} \otimes \mathbf{MEX} \approx \bigoplus_{\ell} (\mathbf{USA}.\ell \otimes \mathbf{MEX}.\ell) : \text{Set}(\text{unordered pairs of values})$$

That type is a **symmetric relation** on values. Bound with a value,  $\mathbf{F}$  returns its partner:

$\mathbf{F} \otimes \text{dollar} \approx \text{peso}$ , and  $\mathbf{F} \otimes \text{peso} \approx \text{dollar}$ .

The type predicts both limits found in Exercise 3.

- **Item 3.** A symmetric relation has no direction, so  $\mathbf{F}$  cannot carry an asymmetric analogy such as *dog : puppy :: cat : kitten*. The direction would have to live in the labels.
- **Item 4.** The pairing needs *the same* record type on both sides. Drop the cur field from one record and dollar has no partner: a type mismatch. A VSA reports a mismatch as noise, never as an error.

# A VSA Is Typed From the Outside

| In the VSA                       | In type theory                                                     |
|----------------------------------|--------------------------------------------------------------------|
| an atom                          | a constant of a base type, Val or Role                             |
| the codebook                     | the signature: every constant, with its type                       |
| cleanup, $cl_T$                  | rounding to the constants of the type the question asks for        |
| binding with a role              | building a field; read semantically, a role $\langle v, e \rangle$ |
| unbinding, then cleanup          | selecting a field, $E.agent$                                       |
| bundling fields; bundling values | forming a record; forming a set                                    |
| $\rho^i$                         | a numbered label: a position in a tuple, or a depth in a tree      |
| $X \otimes Y$ , one record type  | a symmetric relation between their values                          |

The vectors carry no types. Every operation applies to every vector and nothing ever fails: a type error comes back as noise, or, for a path read in the wrong order, as a confident wrong answer.

The type discipline lives in the codebook and in the rules for using the operations, which is another way of putting this deck's closing claim: the symbols live in the codebook and in the rounding.

## In-Class Exercise 4: Types

- 1 Give the type of  $\mathbf{r}_{\text{agent}} \otimes \mathbf{alice} \oplus \mathbf{r}_{\text{theme}} \otimes (\mathbf{book} \oplus \mathbf{pen})$ , and say which kind of cleanup each of its two queries needs.
- 2 In the record for *Alice bought a book from Mary*, the query  $\mathbf{E} \otimes \mathbf{mary}$  asks which role mary fills. Which typing rule covers it, and what is the type of the answer?
- 3 Show that in plain MAP the paths `theme.agent` and `agent.theme` compute the same vector. Then store every embedded record as  $\rho(\mathbf{E})$  and show that they differ.
- 4 ★ Write a typing rule for  $\mathbf{X} \otimes \mathbf{Y}$  when  $\mathbf{X}$  and  $\mathbf{Y}$  are records of one type, and use it to explain why  $\mathbf{F} \otimes \mathbf{dollar}$  fails when one country record has no `cur` field.

Seven minutes. Item 4 is the starred one.

## In-Class Exercise 4: Answers

1.  $\{\text{agent: Val, theme: Set(Val)}\}$ : BIND twice, SET inside the second, RECORD to join. The agent query takes the argmax over the values; the theme query must **threshold**, returning every value above the floor, because its field holds a set.
2. None of the four rules as written: it is SELECT with the label and the value swapped, which MAP allows only because binding is self-inverse. The answer is a role, so it needs  $\text{cl}_{\text{Role}}$ , and it returns  $\mathbf{r}_{\text{source}}$ . The algebra cannot tell a field's name from its value; only the typing can.
3. Plain:  $\mathbf{S} \otimes \mathbf{r}_{\text{th}} \otimes \mathbf{r}_{\text{ag}} = \mathbf{S} \otimes \mathbf{r}_{\text{ag}} \otimes \mathbf{r}_{\text{th}}$ , by commutativity. Protected: the two queries are  $\rho^{-1}(\mathbf{S} \otimes \mathbf{r}_{\text{th}}) \otimes \mathbf{r}_{\text{ag}} = \rho^{-1}(\mathbf{S}) \otimes \rho^{-1}(\mathbf{r}_{\text{th}}) \otimes \mathbf{r}_{\text{ag}}$  and  $\rho^{-1}(\mathbf{S}) \otimes \rho^{-1}(\mathbf{r}_{\text{ag}}) \otimes \mathbf{r}_{\text{th}}$ . They use different keys, since  $\rho^{-1}(\mathbf{r}_{\text{th}}) \otimes \mathbf{r}_{\text{ag}} \neq \rho^{-1}(\mathbf{r}_{\text{ag}}) \otimes \mathbf{r}_{\text{th}}$  for random roles.
4. ★ If  $\mathbf{X}, \mathbf{Y} : \{\vec{\ell} : \text{Val}\}$  with the *same* labels, then  $\mathbf{X} \otimes \mathbf{Y} \approx \bigoplus_{\ell} (\mathbf{X}.\ell \otimes \mathbf{Y}.\ell)$ , of type Set of unordered pairs of values. Without a cur field in  $\mathbf{Y}$  the premise fails: the term  $\mathbf{r}_{\text{cur}} \otimes \mathbf{dollar}$  meets no  $\mathbf{r}_{\text{cur}}$  to cancel against, so it adds only noise, and  $\mathbf{F} \otimes \mathbf{dollar}$  returns nothing above the floor.

## The Families, Briefly

|                      | Vectors         | Bind                 | Note                              |
|----------------------|-----------------|----------------------|-----------------------------------|
| TPR (Smolensky 1990) | real            | outer product        | exact, non-commutative; $n^2$     |
| HRR (Plate 1995)     | real            | circular convolution | fixed $n$ ; needs unitary vectors |
| BSC (Kanerva 1996)   | binary          | XOR                  | self-inverse; majority bundling   |
| MAP (Gayler 1998)    | bipolar         | elementwise product  | what we used today                |
| FHRR                 | complex phasors | phase addition       | HRR in the Fourier domain         |

All five implement the same three operations with the same algebraic laws. The choice is about hardware and exactness, not about expressive power.

They differ in one way that matters for us: **only TPR has a non-commutative binding**, and it pays  $n^2$  for it. Everything that fits in fixed dimension gets order from roles or permutation instead.

The binary variants are attractive for hardware – XOR and majority are a few gates – which is why this family shows up in low-power and in-memory computing work.

# What It Buys, and What It Does Not

| Buys                              | Does not                         |
|-----------------------------------|----------------------------------|
| structure in and structure out    | any learning of the atoms        |
| recursion at constant dimension   | graded similarity between atoms  |
| exact algebraic laws, no training | resolution of lexical ambiguity  |
| analogy by one multiplication     | discovery of an alignment        |
| robustness to coordinate noise    | graceful degradation on overload |

The objection from Part II, now answerable. If atoms carry no semantic similarity, in what sense is this semantics? Answer: it is not a theory of lexical meaning at all. It is a theory of *composition*, and it presupposes that something else gave you the lexicon.

So VSA and distributional semantics are complements, not rivals. One learns what dog means and cannot say who bit whom; the other knows exactly who bit whom and has no idea what a dog is. Seeding the atoms from embeddings is the obvious move, and costs orthogonality, hence capacity.

# VSA Against Attention

Worth comparing directly, since both are bilinear machinery for combining two representations.

|                   | VSA                      | Attention                     |
|-------------------|--------------------------|-------------------------------|
| the middle $G$    | fixed by the algebra     | learned, $W_Q^\top W_K$       |
| symmetric?        | bind and bundle both are | no, and that is the point     |
| where order lives | roles, or $\rho$         | the antisymmetric half of $G$ |
| atoms             | random, fixed            | learned, and they move        |
| decomposable?     | yes, by unbinding        | not in general                |

Read the *symmetric* row against Session 11. A transformer gets argument order from the non-symmetry of a learned  $G$ ; a VSA gets it by stipulating role atoms. Same requirement, two ways of meeting it.

# The Row That Is a Research Question

The last row of that table is the one worth sitting with.

A VSA can take a structure apart **because the structure was put in deliberately**, by someone who kept the codebook. Unbinding works since you know which role to unbind by.

A trained network has no codebook and no declared roles. Whether its representations decompose at all is not known in general, and the evidence is mixed. I would not assert a verdict here.

Two directions people are pushing, both live:

- **Install binding.** Put VSA-style operations inside a trained network, so the structure is guaranteed rather than hoped for. Buys decomposability, costs flexibility.
- **Look for binding.** Probe trained models for role-like directions already present. This is where the attention-head work in our own group sits.

The underlying question is the one Session 11 posed: can a system *learn* the antisymmetric half, or does it have to be given?

## Back to the Question

Last session asked where symbolic reasoning resides in a vector representation. VSA gives the cleanest possible answer, because it is the architecture that does nothing else.

| Capacity            | In a VSA              | Supplied by                      |
|---------------------|-----------------------|----------------------------------|
| identity            | near-orthogonal atoms | dimension, and randomness        |
| structure           | role-filler bindings  | stipulated role atoms, or $\rho$ |
| inference           | unbind, then clean up | the codebook                     |
| <b>discreteness</b> | cl                    | <b>the decision to round</b>     |

The fourth row is the one to take away. The vectors, the operations and the arithmetic are all continuous. The system is symbolic because at chosen moments it *rounds to a finite inventory*.

So “where do the symbols live?” has a location: not the representation and not the algebra, but the codebook plus the rounding step. Remove cleanup and you have an analog device that forgets. A substantive claim, and you may disagree with it.

## Where You Actually See This

VSA is a small field, and it is worth knowing where it has traction and where it has not.

| Area                             | Why it fits                                |               |
|----------------------------------|--------------------------------------------|---------------|
| low-power and in-memory hardware | XOR and majority are cheap; noise-tolerant | real traction |
| robotics, sensorimotor binding   | fixed dimension, fast, online              | real traction |
| cognitive modeling               | an account of systematic composition       | its origin    |
| mainstream NLP                   | atoms carry no meaning                     | marginal      |

Be clear about the last row. No competitive language model is built this way, and the reason is the one from Part II: a representation whose atoms are mutually orthogonal cannot generalize across them.

Everything a language model does well depends on dog and cat being close.

The reason to learn it anyway is that it isolates one capacity perfectly. If you want to know what compositional structure costs and what it requires, this is the system that does nothing else, so nothing else can be taking the credit.

# The Argument This Is Part Of

VSA is one side of a forty-year dispute, and it helps to know which side.

---

|                |                    |                                             |
|----------------|--------------------|---------------------------------------------|
| <b>1988</b>    | Fodor and Pylyshyn | connectionist nets lack systematicity       |
| <b>1990</b>    | Smolensky          | tensor product binding; the reply, at $n^2$ |
| <b>1995</b>    | Plate              | HRR – fold it to fixed $n$                  |
| <b>1996–98</b> | Kanerva, Gayler    | binary and bipolar variants; analogy        |
| <b>2009</b>    | Gayler, Kanerva    | “hyperdimensional computing” as a programme |

---

The claim being defended throughout is that distributed representations *can* be compositional and systematic – that you do not have to choose between vectors and structure. Today’s arithmetic is the demonstration.

What the demonstration does not settle is whether this is how anything *learns*. Every structure we built today was built by hand, from an inventory someone chose. Fodor and Pylyshyn asked whether a network could be systematic; the honest reply is that a network *given* a VSA is systematic, which is a different claim.

# Summary

- 1 A VSA trades learned representations for random ones, and spends the dimension on an alphabet of near-orthogonal atoms. At  $d = 10^4$  nine random atoms had maximum pairwise  $|\cos|$  of 0.025.
- 2 Three operations: **bundle** preserves similarity and makes a set, **bind** destroys similarity and makes a pair, **permute** supplies order.
- 3 **Binding in every fixed-dimension variant is commutative**, so argument order comes from stipulated role atoms or from  $\rho$  – exactly the gap Session 11 identified.
- 4 **Cleanup memory** is not a detail. It restores discreteness, and it is where the system becomes symbolic rather than analog.
- 5 Capacity grows linearly in  $d$  and fails as a cliff. Nesting costs signal at every level.
- 6 Analogy comes out as one multiplication, with no training – but only between structures already aligned role-for-role.
- 7 **Typed from the outside.** The vectors carry no types: the codebook is the signature, cleanup rounds by type, and  $\rho$  keeps paths in order.

Next: the geometric product, which keeps both halves of a bilinear form at once, and so does not have to choose between similarity and order.

# Exercises

- 1 Implement MAP in numpy: random bipolar atoms,  $\otimes$  as  $*$ ,  $\oplus$  as `np.sign(sum)`. Verify that  $\mathbf{a} \otimes \mathbf{a} = \mathbf{1}$  and that  $\cos(\mathbf{a} \otimes \mathbf{b}, \mathbf{a}) \approx 0$ .
- 2 Encode *Alice bought a book from Mary* as a record with four roles. Query every slot, and confirm cleanup returns the right atom each time. Then add four more roles and report when it first fails.
- 3 Reproduce the capacity table for  $d \in \{256, 1000, 10000\}$  and  $k$  up to 400. Plot accuracy against  $k/d$  and say what the collapse point is as a fraction of  $d$ .
- 4 Build the Dollar of Mexico analogy and verify the result. Then break it as in In-Class Exercise 3, item 4, by giving the two records different role sets, and report the cosines.
- 5 Encode the string *abcabc* with  $\rho$ , then query positions 0 through 5. The letter *a* appears twice – explain what the position queries return and why repetition is not a problem here but would be for a bundle.
- 6 ★ Seed the atoms from real 300-dimensional word embeddings instead of randomness. Measure how much capacity you lose, and explain the loss in terms of the cross-terms no longer being orthogonal.