Towards a Formal Distributional Semantics: Simulating Logical Calculi with Tensors

E. Grefenstette

University of Oxford

*SEM 2013

What this paper is about (sort of)

Distributional semantics!

- “You shall know a word by the company it keeps”.
- Words are vectors in high dimensional space.
- Quantitative picture of semantics.

Compositional distributional/distributed semantics!

- Going from word to sentence.
- Defining composition operations for distributed representations.
- Doing cool things with distributed sentence representations.

Tensors in Compositional Distributional Semantics

Tensors are everywhere:

- Baroni and Zamparelli: adjectives are matrices;
- Zanzotto *et al.*: generalised matrix-based vector addition;
- Coecke *et al.* and Grefenstette *et al.*: everything is a tensor;
- Socher *et al.*: recursive matrix-vector models.
- etc.

Why?

Because tensor representations of words give word representations the power of *functions*.

What this paper is about (really)

This paper is about:

- Doing logic with tensors.
- Doing more logic with tensors.

This paper is also about:

- Not doing logic with tensors.

Why you should care

- Not quite sure what the "semantics of distributional semantics" are.
- Relation between distributional accounts and logic is murky.
- Should there be one semantic representation to rule them all?

Spoiler Alert

Don't put all your (semantic) eggs in one (mathematical) basket.

Tensors: a quick and dirty overview

- Order 1 — vector:

$$\overrightarrow{v} \in A = \sum_i C_i^v \ \overrightarrow{a_i}$$

- Order 2 — matrix:

$$M \in A \otimes B = \sum_{ij} C_{ij}^M \ \overrightarrow{a_i} \otimes \overrightarrow{b_j}$$

- Order 3 — cuboid:

$$R \in A \otimes B \otimes C = \sum_{ijk} C_{ijk}^R \ \overrightarrow{a_i} \otimes \overrightarrow{b_j} \otimes \overrightarrow{c_k}$$

Tensor contraction

Tensor contractions:

- Order 1 $\times$ order 1: inner product (dot product)
- Order 2 $\times$ order 1: matrix-vector multiplication
- Order 2 $\times$ order 2: matrix multiplication

Tensor contraction is nothing fancier than a generalisation of these operations to any order.

- Order $n \times$ order m: sum through shared indices.

Order $n \times$ order m contraction yields tensor of order $n + m - 2$.

Tensors as functions

Tensor-linear map isomorphism (Bourbaki, 1985; Lee, 1997)

For any multilinear map $f : V_1 \to \ldots \to V_n$ there is a tensor $T^f \in V_n \otimes \ldots \otimes V_1$ such that for any $\overrightarrow{v_1} \in V_1, \ldots, \overrightarrow{v_{n-1}} \in V_{n-1}$, the following equality holds

$$f(\overrightarrow{v_1}, \ldots, \overrightarrow{v_{n-1}}) = T^f \times \overrightarrow{v_1} \times \ldots \times \overrightarrow{v_{n-1}}$$

Tensors therefore act as functions, with tensor contraction as function application.

Tensors as functions

Properties of linear maps propagate to tensors

- $f \circ g \cong T^f \times T^g$
- $f^{-1} \cong \left(T^f\right)^{-1}$
- $f(\alpha x) = \alpha f(x) \cong \alpha T^f \times T^x = T^f \times \alpha T^x$

Public Service Announcement

Friends don't let friends implement tensors

- `http://www.wlandry.net/Projects/FTensor` (C++)
- `http://www.sandia.gov/~tgkolda/TensorToolbox/` (MATLAB)
- `numpy.array` + `numpy.einsum` (Python)

Cuboid Tensors

A simplified notation for $m \times n \times p$ order-3 tensors (cuboids):

$$T = \left[\begin{array}{ccc|c|ccc} a_{111} & \cdots & a_{1n1} & & a_{11p} & \cdots & a_{1np} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ a_{m11} & \cdots & a_{mn1} & & a_{m1p} & \cdots & a_{mnp} \end{array} \right]$$

Tensor-vector multiplication:

$$T \times \begin{bmatrix} v_1 \\ \vdots \\ v_p \end{bmatrix} = v_1 \begin{bmatrix} a_{111} & \cdots & a_{1n1} \\ \vdots & \ddots & \vdots \\ a_{m11} & \cdots & a_{mn1} \end{bmatrix} + \ldots + v_p \begin{bmatrix} a_{11p} & \cdots & a_{1np} \\ \vdots & \ddots & \vdots \\ a_{m1p} & \cdots & a_{mnp} \end{bmatrix}$$

Simple Logical Models

A logical model $(\mathcal{D}, \mathbb{B}, \{f_{P_i}\}_i, \{f_{R_j}\}_j)$:

- $\mathcal{D}$, the set of logical atoms (domain),
- $\mathbb{B} = \{\top, \bot\}$, the set of truth values,
- $\{f_{P_i} : \mathcal{D} \to \mathbb{B}\}_i$, the set of unary truth functions (predicates),
- $\{f_{R_j} : \mathcal{D} \times \mathcal{D} \to \mathbb{B}\}_j$, the set of binary truth functions (binary relations).

Logical Models with Tensors

Representing $(\mathcal{D}, \mathbb{B}, \{f_{P_i}\}_i, \{f_{R_j}\}_j)$ with tensors:

- The set of one-hot vectors in $D \cong \mathbb{R}^{size(\mathcal{D})}$ models the logical atoms of $\mathcal{D}$. For example:

$$\mathcal{D} = \{a, b, c\} \;\Rightarrow\; \mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \; \mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \; \mathbf{c} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

- $\top$ and $\perp$ form a basis of $B \cong \mathbb{R}^2$:

$$\top = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \perp = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Logical Models with Tensors

Unary truth functions $f_{P_i} : \mathcal{D} \to \mathbb{B}$ translate to linear maps $f'_{P_i} : D \to B$, which we represent as tensors $\mathbf{M}_{P_i} \in B \otimes D$.

$$\text{E.g. } f_P(x) = \begin{cases} \top \text{ if } x \in \{a, b\} \\ \bot \text{ otherwise} \end{cases} \Rightarrow \mathbf{M}_P = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$Pa = \mathbf{M}_P \times \mathbf{a} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \top$$

Examples

Binary truth functions $f_{R_j} : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{B}$ become linear maps $f'_{R_j} : D \times D \rightarrow B$, represented as tensors $\mathbf{M}_{R_j} \in B \otimes D \otimes D$.

$$\text{E.g. } f_R(x, y) = \begin{cases} \top \text{ if } (x, y) \in \{(a,\ c),\ (b,\ b)\} \\ \bot \text{ otherwise} \end{cases}$$

$$\Rightarrow\ \mathbf{M}_R = \left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$Rba = (\mathbf{M}_R \times \mathbf{b}) \times \mathbf{a}$$

$$= \left(\left[\begin{array}{ccc|ccc|ccc} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}\right] \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \bot$$

Logical Operations with Tensors

Logical operations: $f_{op} : B \times B \to B \Rightarrow \mathbf{T}^{op} : B \otimes B \otimes B$.

Step 1: Express truth tables in unary component form by fixing truth value of first argument.

Unary components:

$$x \mapsto x : \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad x \mapsto \neg x : \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$x \mapsto \top : \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \qquad x \mapsto \bot : \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Example:

a	b	$a \wedge b$
$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$
$\bot$	$\top$	$\bot$
$\bot$	$\bot$	$\bot$

$\Rightarrow$

a	$a \wedge b$
$\top$	$b \mapsto b$
$\bot$	$b \mapsto \bot$

Logical Operations with Tensors

Step 2: Combine both unary components into a cuboid.

In our example, to produce the cuboid tensor for conjunction $\mathbf{T}^{\wedge}$:

$$\begin{array}{c|c} a & a \wedge b \\ \hline \top & b \mapsto b \\ \bot & b \mapsto \bot \end{array} \quad \Rightarrow \quad \mathbf{T}^{\wedge} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array} \right]$$

If the operation is unary (e.g. negation), it is trivially equivalent to its unary component.

Logical Operations with Tensors

We can verify that the truth table is reproduced:

$$\mathbf{T}^{\wedge} \times \top = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{T}^{\wedge} \times \bot = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \top = \top \quad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \times \bot = \bot$$

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \times \top = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \times \bot = \bot$$

a	b	$a \wedge b$
$\top$	$\top$	$(\mathbf{T}^{\wedge} \times \top) \times \top = \top$
$\top$	$\bot$	$(\mathbf{T}^{\wedge} \times \top) \times \bot = \bot$
$\bot$	$\top$	$(\mathbf{T}^{\wedge} \times \bot) \times \top = \bot$
$\bot$	$\bot$	$(\mathbf{T}^{\wedge} \times \bot) \times \bot = \bot$

Logical Operations with Tensors

The full set of connectives:

$$(\neg) \mapsto \mathbf{T}^{\neg} = \left[\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}\right]$$

$$(\vee) \mapsto \mathbf{T}^{\vee} = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

$$(\wedge) \mapsto \mathbf{T}^{\wedge} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$(\rightarrow) \mapsto \mathbf{T}^{\rightarrow} = \left[\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

Logical Operations with Tensors

Some general properties:

- Formalism works for countably infinite domains.
- Predicates, relations, truth values and domain individuals can be probabilistic. For example:

$$\textbf{Ed\_is\_awake} = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix} \quad \textbf{spartacus} = \begin{bmatrix} 0.2 \\ 0.5 \\ 0.3 \end{bmatrix}$$

$$\mathbf{M}_R = \left[\begin{array}{ccc|ccc|ccc} 0.3 & 0.0 & 0.9 & 1.0 & 1.0 & 0.5 & 0.7 & 0.2 & 0.0 \\ 0.7 & 1.0 & 0.1 & 0.0 & 0.0 & 0.5 & 0.3 & 0.8 & 1.0 \end{array}\right]$$

- Probability normalisation is conserved by logical connectives.

Initial Limitations

This tensor-based approach does not support quantifiers:

- To quantify, we need to talk about sets of atoms.
- Semantic representations are reduced to truth values.
- No “book-keeping”.
- We need to track which predicates apply to which atoms.

Overcoming Limitations

We can solve some of these problems by defining a second tensor logic:

- The sum of a set of domain element vectors represents the set of those domain elements. For example:

$$\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mathbf{c} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \Rightarrow \quad \{\mathbf{a}, \mathbf{c}\} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

- We define set union and intersection using non-linear component-wise *min* and *max* maps. For sets U and V modelled as vectors $\mathbf{u}$ and $\mathbf{v}$:

$$U \cap V \Rightarrow min(\mathbf{u}, \mathbf{v}) \qquad U \cup V \Rightarrow max(\mathbf{u}, \mathbf{v})$$

- Union and intersection model disjunction and conjunction over set-vectors, respectively.

Overcoming Limitations

- Predicates and relations become filters, modelling functions $f_P : \mathcal{P}(\mathcal{D}) \to \mathcal{P}(\mathcal{D})$:

$$\mathbf{M}_P = \begin{bmatrix} a & b & c \\ 1-a & 1-b & 1-c \end{bmatrix} \quad \Rightarrow \quad \mathbf{F}_P = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$

- Applying a filter $\mathbf{F}_P$ to a set $\mathbf{s}$ returns the subset $\mathbf{F}_P \times \mathbf{s}$ for which the predicate holds.

Defining Quantifiers

- We define quantifiers as follows:
  - Bound variables, e.g. $\mathbf{x}$ are the vector $\mathbf{1}$.
  - Let $\mathbf{X}$ and $\mathbf{Y}$ be sets obtained by composition (e.g. $\mathbf{X} = \mathbf{F}_P \times \mathbf{x}$).
  - Universal quantification:

$$\forall x.(X \rightarrow Y) \quad \Rightarrow \quad \mathit{forall}(\mathbf{X}, \mathbf{Y}) = \begin{cases} \top & \text{if } \mathbf{X} = \mathit{min}(\mathbf{X}, \mathbf{Y}) \\ \bot & \text{otherwise} \end{cases}$$

  - Existential quantification:

$$\exists x.(X) \quad \Rightarrow \quad \mathit{exists}(\mathbf{X}) = \begin{cases} \top & \text{if } |\mathbf{X}| > 0 \\ \bot & \text{otherwise} \end{cases}$$

A Tale of Two Tensor Logics

Pros:

- Both of these approaches are related.
- Can go back and forth between predicate and filter tensors.
- Together, they simulate a fairly full predicate logic.

Cons:

- Can't use both approaches at the same time.
- Quantifiers require non-linearity.
- No scope.
- Models, not syntactic inference.

Conclusions

- You can simulate quite a lot of logic with tensors.
- Nice properties for some kinds of probabilistic logics.
- There are some aspects of logic tensor models don't capture well.
- Not everything can be done in *just* the distributional setting:
  - Make use of non-linearities?
  - Use distributional semantics in addition to "real" logical models?
  - Stick around for the next talk. . .

Thank you for listening!