Algebraic Approaches to Compositional Distributional Semantics

Daoud Clarke[1], David Weir[2] and Rudi Lutz[2]
[1] Department of Computer Science, University of Hertfordshire
[2] Department of Informatics, University of Sussex

The question of how to compose meaning in distributional representations of meaning has recently been recognised as a central issue in computational linguistics. We describe three general and powerful tools that can be used to describe composition in distributional semantics: quotient algebras, learning of finite dimensional algebras, and the construction of algebras from semigroups.

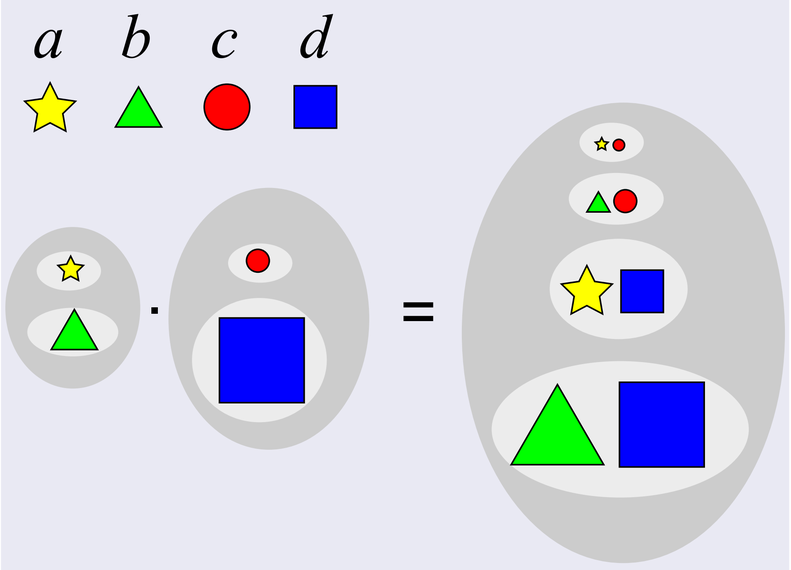
Algebra Over a Field

A real algebra $\mathcal{A}$ is a **vector space** over the real numbers together with an **associative bilinear multiplication**:

$$x(y+z) = xy + xz$$
$$(x+y)z = xz + yz$$
$$(\alpha x)(\beta y) = \alpha\beta xy$$
$$(xy)z = x(yz)$$

where $x, y, z \in \mathcal{A}$ and $\alpha, \beta \in \mathbb{R}$.

Example:



- Graphical depiction of

$$(0.5a + b) \cdot (0.5c + 2d)$$
$$= 0.25ac + 0.5bc + ad + 2bd$$

- **Order n matrices** form an algebra:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$$

Learning Finite Algebras

Idea:

- Generate vectors for words and pairs of words using e.g. **latent semantic analysis**
- Learn the product that best fits these vectors using e.g. **least squares**
- Test on a sample of the ukWaC corpus.
- Extract a list of $verb\ adjective^*noun$ sequences

Advantages:

- Composition defined by data

Disadvantages:

- Long range compositionality must be compromised because of finite dimensionality
- We have so far been unable to obtain good results using a least squares regression method

Quotient Algebras

The idea:

- Construct a free (tensor) algebra:

$$T(V) = \mathbb{R} \oplus V \oplus (V \otimes V) \oplus (V \otimes V \otimes V) \oplus \cdots$$

- Impose relations by taking quotients
- Choose a set of pairs of vectors which should be equal in the quotient algebra
- Put their differences into a **generating set** Λ
- Find the minimal ideal containing Λ (the ideal **generated** by Λ)
- Take the **quotient vector space** $T(V)/I$
- This is a vector space formed from equivalence classes defined by

$$x \equiv y \text{ if } x - y \in I$$

Data-driven Composition

- In the tensor algebra, strings of different length are **not comparable** (their inner product is always zero)
- Choose elements of Λ based on a **treebank**, or parsed corpus data
- Strings of the same **grammatical type** become comparable in the quotient algebra

Quotient Algebras Example

Use a small corpus:

see red apple
buy apple
read big book
throw old small red book
buy large new book
see big city
visit big apple
modernise city
see modern city

Results (cosine similarity values):

	apple	*big apple*	*red apple*	*city*	*big city*	*red city*	*book*	*big book*	*red book*
apple	1.0	0.26	0.24	0.52	0.13	0.12	0.33	0.086	0.080
big apple		1.0	0.33	0.13	0.52	0.17	0.086	0.33	0.11
red apple			1.0	0.12	0.17	0.52	0.080	0.11	0.33
city				1.0	0.26	0.24	0.0	0.0	0.0
big city					1.0	0.33	0.0	0.0	0.0
red city						1.0	0.0	0.0	0.0
book							1.0	0.26	0.24
big book								1.0	0.33
red book									1.0

- Strings of different lengths have non-zero inner product
- Similarity of *red apple* and *red city* is the same as that of *apple* and *city*
- Many properties of the tensor product carry over to the quotient algebra

Construction from Semigroups

Problems:

- Previous two techniques do not easily permit **complex semantics**
- Negation?
- Passive vs. active?

These problems have been solved in e.g. **Montague semantics** or **Discourse Representation Theory**.

- Take existing semantic framework and "vectorize" it
- Avoid redoing all existing work in semantics but in a vector form
- Logic $\Rightarrow$ Semigroup $\Rightarrow$ Algebra

Semigroup Example

Example context vectors:

Term	Context vector
fish	$(0, 0, 1)$
big	$(1, 2, 0)$

Noun and adjective definitions:

$$n_i = (N, \lambda x\ \text{noun}_i(x))$$
$$a_i = (N/N, \lambda p \lambda y\ \text{adj}_i(y) \wedge p.y)$$

- Words are pairs (s, σ)
- s describes the syntactic type (e.g. **Lambek calculus**)
- σ describes semantics (first order logic and lambda calculus)
- Adjective and noun components compose as follows:

$$a_i n_j = (N, \lambda x(\text{noun}_j(x) \wedge \text{adj}_i(x)))$$

- Define vectors for *big* and *fish* by

$$\widehat{big} = a_1 + 2a_2 \qquad \widehat{fish} = n_3$$

- Then *big fish* has the vector

$$\widehat{big}\ \widehat{fish} = a_1 n_3 + 2a_2 n_3$$
$$= (N, \lambda x(\text{noun}_3(x) \wedge \text{adj}_1(x))) + 2(N, \lambda x(\text{noun}_3(x) \wedge \text{adj}_2(x)))$$

Problems:

- Elements a_i form a **commutative, idempotent** subsemigroup of S
- This means they have a **semilattice** structure describing **entailment**
- This entailment relation is not encoded in the vector space structure
- Perhaps there is a more sophisticated construction (e.g. C* enveloping algebras) that could accommodate this.