Compositional Models of Vector-based Semantics: From Theory to Tractable Implementation

Day 2: The Curse of Dimensionality

Gijs Wijnholds & Michael Moortgat

ESSLLI 2022

Abstract

Vector-based compositional architectures combine a distributional view of word meanings with a modelling of the syntax-semantics interface as a structure-preserving map relating syntactic categories (types) and derivations to their counterparts in a corresponding meaning algebra.

This design is theoretically attractive, but faces challenges when it comes to large-scale practical applications. First there is the curse of dimensionality resulting from the fact that semantic spaces directly reflect the complexity of the types of the syntactic front end. Secondly, modelling of the meaning algebra in terms of finite dimensional vector spaces and linear maps means that vital information encoded in syntactic derivations is lost in translation.

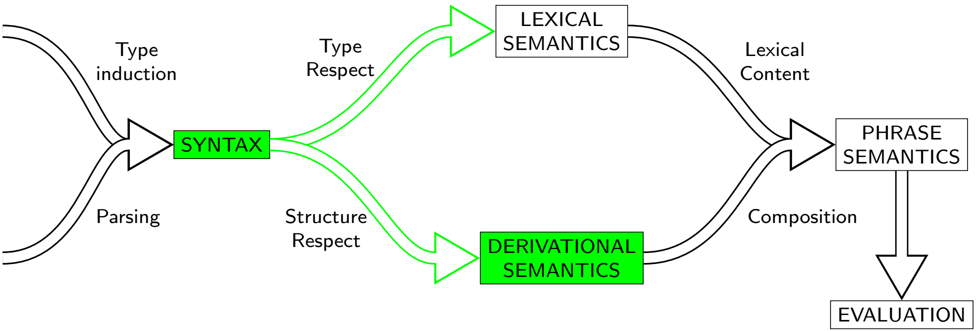
The course compares and evaluates methods that are being proposed to face these challenges. Participants gain a thorough understanding of theoretical and practical issues involved, and acquire hands-on experience with a set of user-friendly tools and resources.

Today: the curse of dimensionality

Day plan

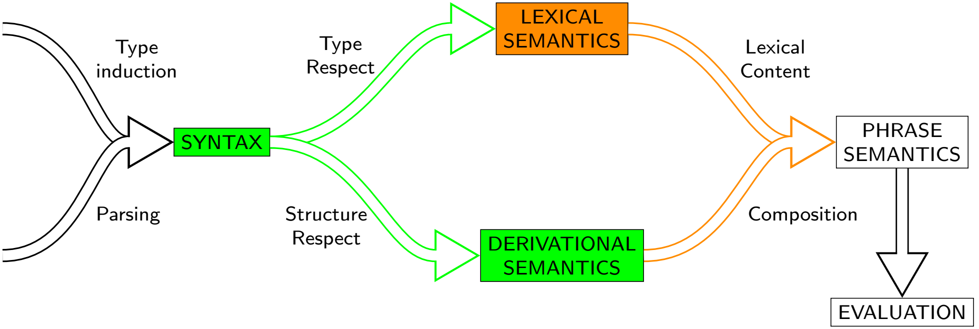
- The curse of dimensionality in tensor-based compositional models: naive models require large tensor representations
- Lexical semantics for great expressivity: toning down the size of function word tensors
- The curse of dimensionality in representation learning: skipgram matrices, dependencies as matrices, tensor decompositions
- Where do we go from here?

Recap: The Compositional Process



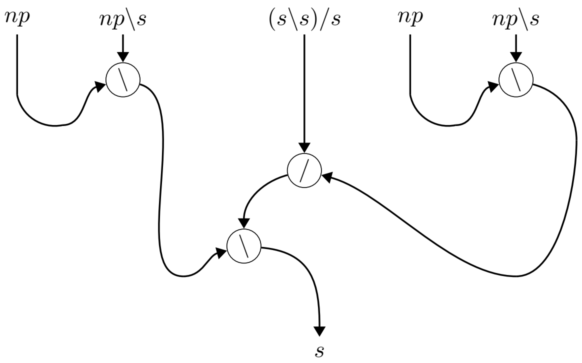
- Our core methodology provides syntax and the interfacing with semantics,
- Lexical content is learnable, though not always in a tractable way (Tue)
- Syntax doesn't come for free: type induction & parsing as learnable processes (Wed/Thu)
- In the end, the phrase semantics can be applied to NLP tasks (Fri)

Today: The Compositional Process



- We discuss the consequences of the basic methodology for lexical semantics,
- Lexical content is learnable, though not always in a tractable way,
- We look at ways to mitigate the curse of dimensionality,
- Composition may be left open, i.e. we are not bound to a linear map interpretation of semantic composition.

Diagrams Galore

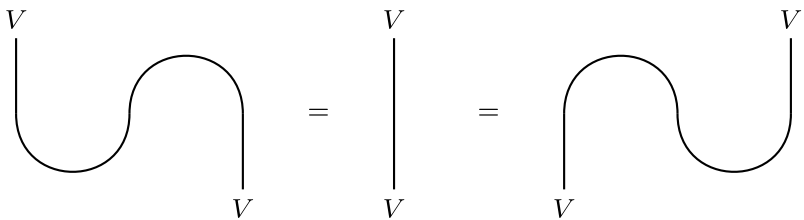


Diagrams for sCCCs + Frob

Diagrammatic reasoning The contraction/expansion operations on finite-dimensional vector spaces can be depicted visually:

$$\epsilon : V \otimes V \to I \qquad \eta : I \to V \otimes V$$

with the according 'yanking' equations fulfilling the sCCC closure property:



Syntax Diagrams

[Wijnholds, 2017] considers categorical proof nets: graphical representations of the deductions in the Lambek Calculus that also satisfy the categorical axioms of a biclosed monoidal category. In short, one extends the language of string diagrams with links for each connective in the calculus (top: destructor links, bottom: constructor links):

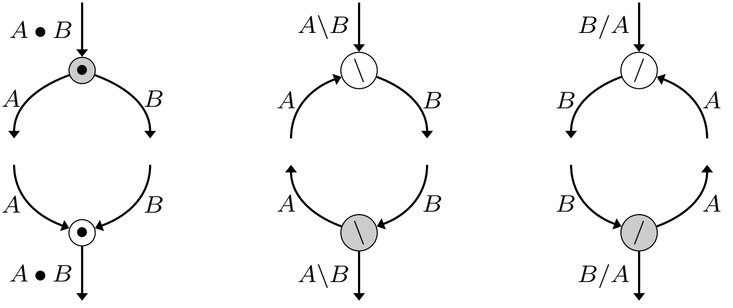
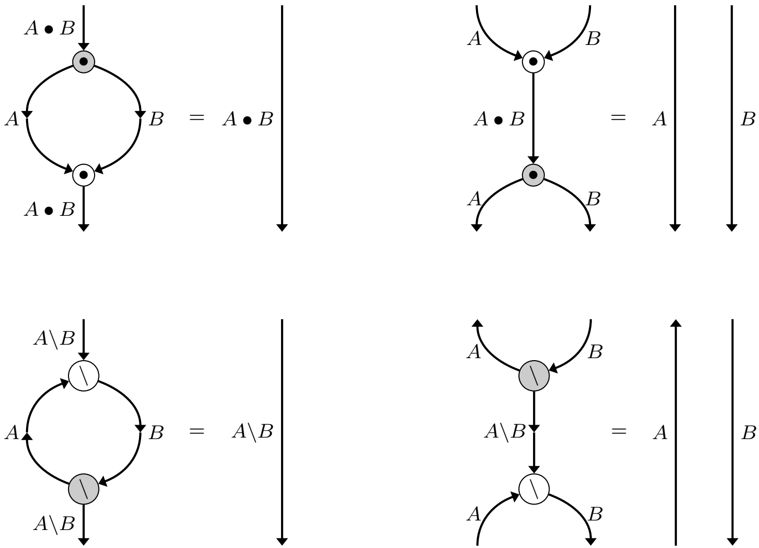


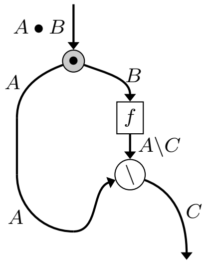
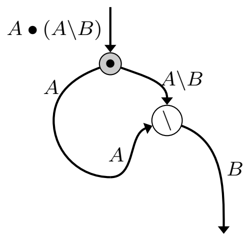
Diagram Equations

Cut! Attaching a destructor link to a constructor link in either order gives a structure that may be cut out of a diagram:



Residuation and application

Combinators The application law $A \otimes A\backslash B \to B$ can be expressed with the combinator $\lhd^{-1} 1_{A\backslash B}$. Visually:

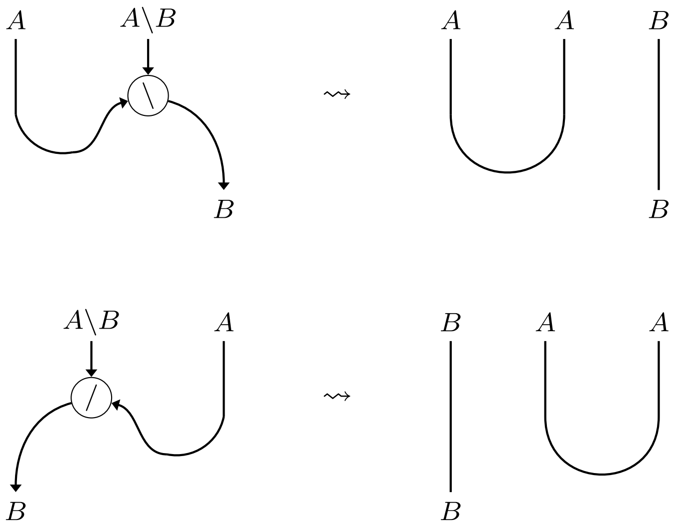
Residuation $\lhd^{-1}$	Application
	

Brain Teaser Prove the below equations visually:

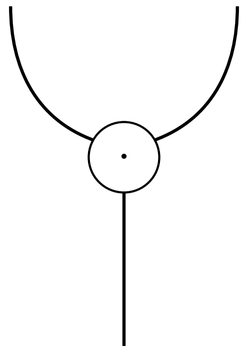
- Bifunctoriality of $\bullet$: $(k \bullet h) \circ (g \bullet f) = (k \circ g) \bullet (h \circ f)$,
  for $g : A \to C, f : B \to D, k : C \to E, h : D \to F$
- Naturality of residuation: $h \circ \lhd^{-1} k \circ (f \otimes g) = \lhd^{-1}((f\backslash h) \circ k \circ g)$,
  for $f : A \to A', g : B \to B', h : C \to C', k : B' \to A'\backslash C$

From syntax to semantics

Interpreting categories The elimination steps in a natural deduction proof that we often use directly translate to sCCC diagrams:

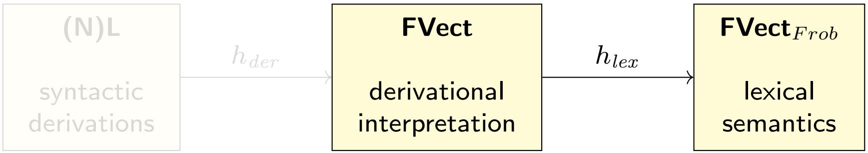


Lexical recipes



Completing the interpretation

Compositional interpretion as a two-step translation h_{der}, h_{lex}:



- **NL** $\simeq$ directional linear λ terms

  source language: syntactic calculus

- **FVect** $=$ finite-dimensional vector spaces and linear maps

  skeleton for meaning assembly, parametric w.r.t. word meaning

- **FVect**$_{Frob}$: we expand **FVect** with Frobenius algebras, which allow us to 'duplicate' information.

Frobenius!

Definition A Frobenius algebra $(X, \Delta, \iota, \mu, \zeta)$ is an object (read: vector space) X with four maps

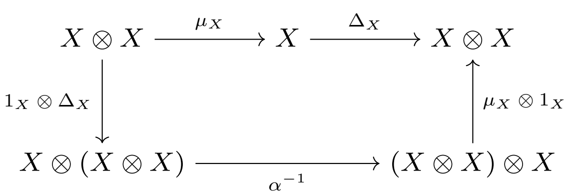
$$\text{(coass.)} \qquad \Delta\colon X \to X \otimes X \qquad \iota\colon X \to I$$

$$\text{(ass.)} \qquad \mu\colon X \otimes X \to X \qquad \zeta\colon I \to X$$

where Δ and μ must comply with the Frobenius condition:

$$(\mu \otimes 1_X) \circ (1_X \otimes \Delta) \;=\; \Delta \circ \mu \;=\; (1_X \otimes \mu) \circ (\Delta \otimes 1_X)$$

I've seen this before... We recognise Δ as a duplicator for information and μ as a merger of information. The Frobenius condition visually:



Frobenius Concretely 1

Duplication For any V we have $\Delta_V : V \to V \otimes V$:

$$\Delta_V\left(\sum_i c_i \vec{v}_i\right) = \sum_i c_i (\vec{v}_i \otimes \vec{v}_i) = \rho_{ijk}\mathbf{v}_k \qquad \rho_{ijk} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{otherwise} \end{cases}$$

Embedding a vector on the diagonal of a matrix:

$$\begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \mapsto \begin{pmatrix} 3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix}$$

Deletion For any V we have $\iota_V : V \to \mathbb{R}$, given by

$$\iota_V\left(\sum_i c_i \vec{v}_i\right) = \sum_i c_i = \tau_i \mathbf{v}_i \qquad \tau_i = \begin{cases} 1 & \text{if } i = i \\ 0 & \text{otherwise} \end{cases}$$

Summing the elements of a vector:

$$\begin{pmatrix} 3 \\ 4 \\ 5 \end{pmatrix} \mapsto 3 + 4 + 5 = 12$$

Frobenius Concretely 2

Merging For any V we have $\mu_V : V \otimes V \to V$:

$$\mu_V\left(\sum_{ij} c_{ij}(\vec{v}_i \otimes \vec{v}_j)\right) = \sum_i c_{ii}\vec{v}_i = \rho_{ijk}\mathbf{M}_{ij} \qquad \rho_{ijk} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{otherwise} \end{cases}$$

Retrieve the diagonal of a matrix:

$$\begin{pmatrix} 6 & 1 & 5 \\ 3 & -9 & -4 \\ -2 & 10 & 7 \end{pmatrix} \mapsto \begin{pmatrix} 6 \\ -9 \\ 7 \end{pmatrix}$$

Insertion For any V we have $\zeta_V : \mathbb{R} \to V$, given by:

$$\zeta_V\left(\lambda\right) = \sum_i \lambda\vec{v}_i = \lambda\tau_i \qquad \tau_i = \begin{cases} 1 & \text{if } i = i \\ 0 & \text{otherwise} \end{cases}$$

Embedding a number in a vector:

$$\lambda \mapsto \begin{pmatrix} \lambda \\ \lambda \\ \lambda \end{pmatrix}$$

Brain Teaser Time

Frobenius condition

$$(\mu_V \otimes 1_V) \circ (1_V \otimes \Delta_V) \;=\; \Delta_V \circ \mu_V \;=\; (1_V \otimes \mu) \circ (\Delta \otimes 1_V)$$

Speciality

$$\mu_V \circ \Delta_V = 1_V$$

(Try it out : $\mathbf{v}_i \mapsto \rho_{jki}\mathbf{v}_i \mapsto \rho_{jkl}\rho_{jki}\mathbf{v}_i$)

Decomposition

$$\varepsilon_V = \iota_V \circ \mu_V \qquad \eta_V = \Delta_V \circ \zeta_V$$

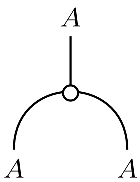
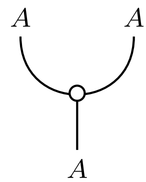
(Because: $\tau_k \rho_{ijk}\mathbf{M}_{ij} = \mathbf{M}_{ii}$) (Because: $\rho_{ijk}\tau_k = \delta_{ij}$)

Tensor Product

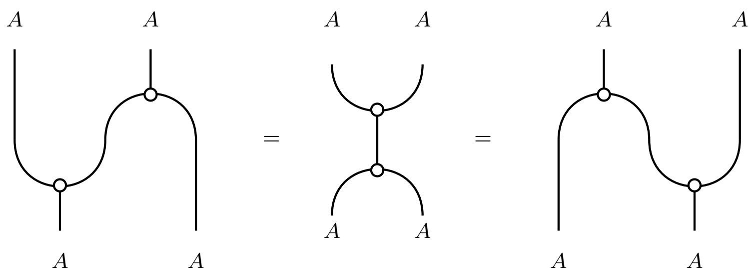
$$\Delta_{V\otimes W} = \Delta_V \otimes \Delta_W$$

$$\mu_{V\otimes W} = (\mu_V \otimes \mu_W) \circ (1_V \otimes \sigma_{W,V} \otimes 1_W)$$

Diagrams and visual equations for Frobenius Algebras

$\Delta : A \to A \otimes A$	$\iota : A \to I$	$\mu : A \otimes A \to A$	$\zeta : I \to A$
			

Frobenius Condition



Coördinators

Recap For chameleon words ('and', 'or', 'not') we may use polymorphic types, making them combine with different elements: $(X\backslash X)/X$ for the binary case, X/X for negation. In formal semantics the binary case is sometimes referred to as generalised coordination with semantic types $A_1 \to ... \to A_n \to t$.

Issue what if we want to coordinate entities, as in 'Bob and Alice'? We'd get $e \to e \to e$ and :

$$\textsc{Bob} \wedge \textsc{Alice} = ???$$

Frobenius! Using μ we can even compute 'intersection' at the vector level:

$$\textsc{Bob} \wedge \textsc{Alice} = \overrightarrow{bob} \odot \overrightarrow{alice}$$

Intuïtion All those features that hold for both Bob and Alice (...)

General We can coordinate any type: given $\overrightarrow{A}, \overrightarrow{B}$ in vector space V, the coordination becomes

$$\overrightarrow{A} \sqcap_{Frob} \overrightarrow{B} = \mu_V(\overrightarrow{A} \otimes \overrightarrow{B}) = \rho_{IJK} A_I B_J$$

Pronoun Relativisation

Subj relative ‘paper that disappointed Bob’ $\rightsquigarrow (n\backslash n)/(np\backslash s)$

Obj relative ‘paper that Bob rejected’ $\rightsquigarrow (n\backslash n)/(s/np)$

Intuition In both cases we’d like to express an intersective meaning (here: obj. relative):

$$\overrightarrow{paper} \odot \iota_S(\overline{\overline{rejected}} \times \overrightarrow{bob})$$

		Syn. type	Vector space
(subj)	that	$(n\backslash n)/(np\backslash s)$	$\mathsf{N}^* \otimes \mathsf{N} \otimes \mathsf{S}^* \otimes \mathsf{N}$
(obj)	that	$(n\backslash n)/(s/np)$	$\mathsf{N}^* \otimes \mathsf{N} \otimes \mathsf{N} \otimes \mathsf{S}^*$

Compare with the formal semantics version:

$$\lambda x^!.((\textsc{paper}\ x) \wedge ((\textsc{rejected}\ x)\ \textsc{Bob})) : !\,e \multimap t$$

In Dutch ‘ring die Frodo vernietigt’; one type, two readings $\rightsquigarrow$ Day 3

Example: Object Relative

Natural Deduction

$$\dfrac{\text{paper}}{n_0} \qquad \dfrac{\text{that}}{(n_1\backslash n_2)/(s_3/np_4)} \qquad \dfrac{\text{Bob}}{np_5} \qquad \dfrac{\text{rejected}}{(np_6\backslash s_7)/np_8} \qquad [np_a \vdash np_a]^1$$

$$\dfrac{(np_6\backslash s_7)/np_8 \qquad [np_a \vdash np_a]^1}{\text{rejected} \cdot np_8 \vdash np_6\backslash s_7}\ [/E]$$

$$\dfrac{np_5 \qquad \text{rejected} \cdot np_8 \vdash np_6\backslash s_7}{\text{Bob} \cdot (\text{rejected} \cdot np_8) \vdash s_7}\ [\backslash E]$$

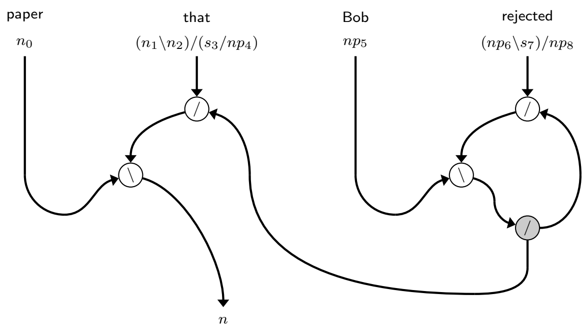
$$\dfrac{\text{Bob} \cdot (\text{rejected} \cdot np_8) \vdash s_7}{(\text{Bob} \cdot \text{rejected}) \cdot np_8 \vdash s_7}\ [Ar]$$

$$\dfrac{(\text{Bob} \cdot \text{rejected}) \cdot np_8 \vdash s_7}{\text{Bob} \cdot \text{rejected} \vdash s_7/np_8}\ [/I]^1$$

$$\dfrac{(n_1\backslash n_2)/(s_3/np_4) \qquad \text{Bob} \cdot \text{rejected} \vdash s_7/np_8}{\text{that} \cdot (\text{Bob} \cdot \text{rejected}) \vdash n_1\backslash n_2}\ [/E]$$

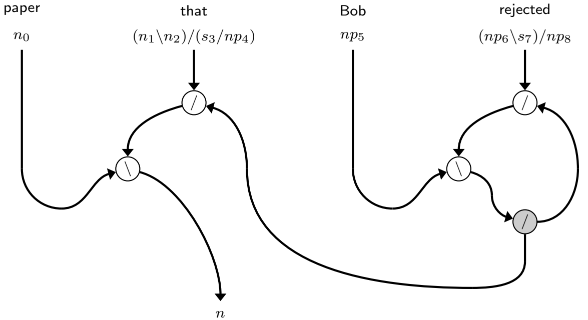
$$\dfrac{n_0 \qquad \text{that} \cdot (\text{Bob} \cdot \text{rejected}) \vdash n_1\backslash n_2}{\text{paper} \cdot (\text{that} \cdot (\text{Bob} \cdot \text{rejected})) \vdash n_2}\ [\backslash E]$$

Diagrammatic

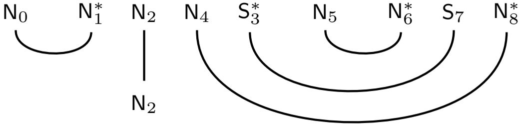


Example: Object Relative

Syntactic Diagram

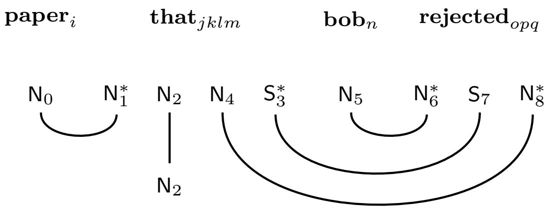


Semantic Diagram

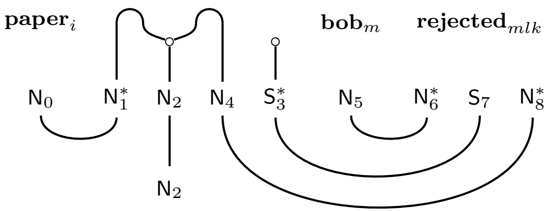


Example: Object Relative

Semantic Diagram with lexical items:



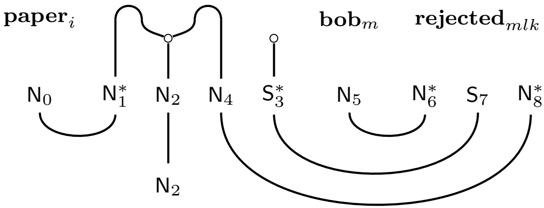
Semantic Diagram after lexical insertion:



$$\mathbf{n}_j = \mathbf{paper}_i \otimes \rho_{ijk}\tau_l \otimes \mathbf{bob}_m \otimes \mathbf{rejected}_{mlk}$$

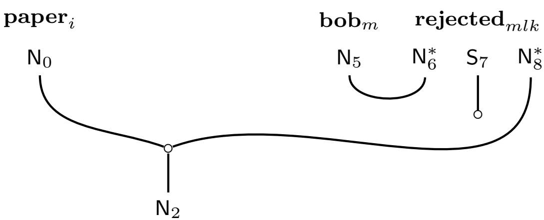
Example: Object Relative

Semantic Diagram after lexical insertion:



$$\mathbf{n}_j = \mathbf{paper}_i \otimes \rho_{ijk}\tau_l \otimes \mathbf{bob}_m \otimes \mathbf{rejected}_{mlk}$$

Semantic Diagram after rewriting:



$$\mathbf{n}_j = \mathbf{paper}_i \odot (\mathbf{bob}_m \tau_l \mathbf{rejected}_{mlk})$$

Mind the gaps...

Parasitic gaps a gap that is felicitous only in the presence of a primary gap. Compare the below:

a	papers that Bob rejected ␣ (immediately)
b	papers that Bob rejected ␣ without reading ␣$_p$ (carefully)

Polymorphism we type the coordinator ‘without’ in a structured way. We instantiate the *polymorphic* type $(X\backslash X)/Y$ with $X = np\backslash s, Y = gp$. We obtain the desired copying behaviour of the gap via a derived type, in two steps:

$$(X\backslash X)/Y \xrightarrow{\text{expand}} ((X\backslash X)/np)/(Y/np) \xrightarrow{\text{distribute}} ((X/np)\backslash(X/np))/(Y/np)$$

Natural deduction

$$\frac{\dfrac{\text{without}}{(((np\backslash s)/np)\backslash((np\backslash s)/np))/(gp/np)} \qquad \dfrac{\text{reading}}{gp/np}}{\text{without} \cdot \text{reading} \vdash ((np\backslash s)/np)\backslash((np\backslash s)/np)}\ [/E]$$

$$\frac{\dfrac{\text{rejected}}{(np\backslash s)/np} \qquad \text{without} \cdot \text{reading} \vdash ((np\backslash s)/np)\backslash((np\backslash s)/np)}{\text{rejected} \cdot (\text{without} \cdot \text{reading}) \vdash (np\backslash s)/np}\ [\backslash E]$$

$$\frac{\text{rejected} \cdot (\text{without} \cdot \text{reading}) \vdash (np\backslash s)/np \qquad [np \vdash np]^1}{(\text{rejected} \cdot (\text{without} \cdot \text{reading})) \cdot np \vdash np\backslash s}\ [/E]$$

$$\frac{\dfrac{\text{Bob}}{np} \qquad (\text{rejected} \cdot (\text{without} \cdot \text{reading})) \cdot np \vdash np\backslash s}{\text{Bob} \cdot ((\text{rejected} \cdot (\text{without} \cdot \text{reading})) \cdot np) \vdash s}\ [\backslash E]$$

$$\frac{\text{Bob} \cdot ((\text{rejected} \cdot (\text{without} \cdot \text{reading})) \cdot np) \vdash s}{(\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading}))) \cdot np \vdash s}\ [Ar]$$

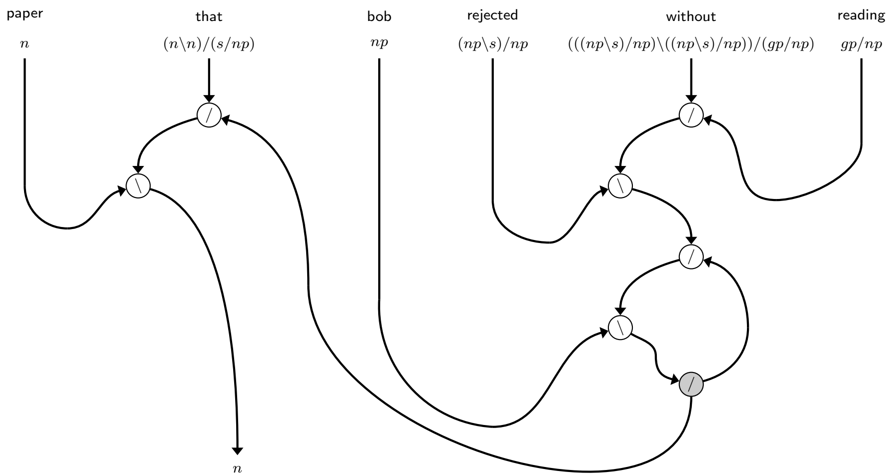
$$\frac{(\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading}))) \cdot np \vdash s}{\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading})) \vdash s/np}\ [/I]^1$$

$$\frac{\dfrac{\text{that}}{(n\backslash n)/(s/np)} \qquad \text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading})) \vdash s/np}{\text{that} \cdot (\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading}))) \vdash n\backslash n}\ [/E]$$

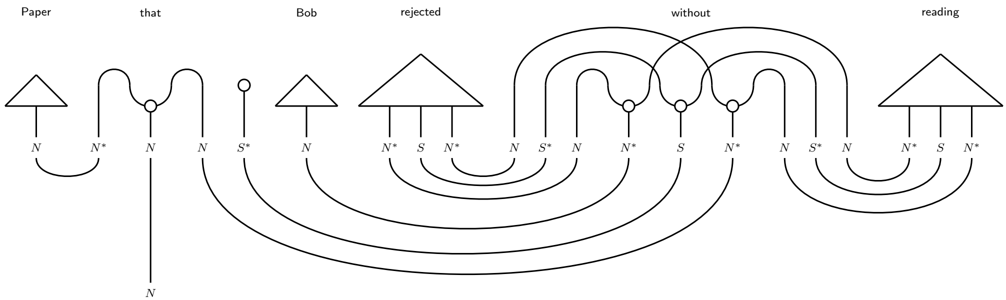
$$\frac{\dfrac{\text{paper}}{n} \qquad \text{that} \cdot (\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading}))) \vdash n\backslash n}{\text{paper} \cdot (\text{that} \cdot (\text{Bob} \cdot (\text{rejected} \cdot (\text{without} \cdot \text{reading})))) \vdash n}\ [\backslash E]$$

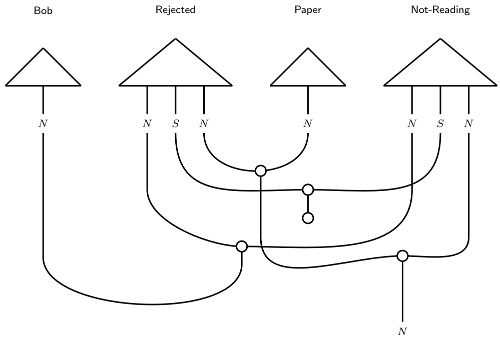
Mind the gaps...

Diagrammatic parasitic the use of hypothetical reasoning disappears in the visual format, associativity becomes apparent from the inbalance between tensor (white) and par (grey) nodes.



Mind the gaps...

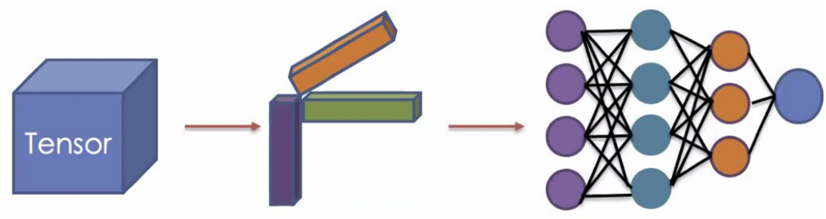




Discussion

- We have seen how variable reuse in lambda terms to model function words, can be analogously modelled in vector-based semantics, using Frobenius Algebras.
- In the vector-based setting, the use of Frobenius Algebras is non-linear (in the variable duplication sense) but linear in the Linear Algebra sense: linear non-linearity!
- A question: how far can the use of Frobenius Algebras for function words take us in reducing the size of tensor representations that we need to learn?
- And what about content words? How do we represent the content of adjectives, verbs, etc? Are current ML techniques adequate for representing these as tensors?

Representation Learning in higher dimensions



The Shape of the Lexicon

- Given our theoretical setup, we require that lexical items adhere to the interpretation of the syntactic type they were assigned.
- A word with syntactic type np, will be interpreted in the noun space N; a word with a complex syntactic type, viz. an intransitive verb, is interpreted as $\lceil np \backslash s \rceil = \mathsf{N} \otimes \mathsf{S}$.
- The consequence is that we need to figure out how to represent words with complex types as higher-order tensors
- Nouns are vectors, but adjectives, verbs, coordinators, etc?
- The process of inducing the appropriate tensors for given words is called representation learning.

Nouns are Vectors, Adjectives are Matrices

Following the compositional distributional methodology, Baroni and Zamparelli [2010] investigates a way of learning adjective matrices using linear regression:

1. Take co-occurrence based vectors $\overrightarrow{n}$ for nouns (normalized using mutual information),
2. Compute adjective-noun vectors $\overrightarrow{An}$ by considering the combo as a single word,
3. Given adjective A, we optimise over the nouns n that occurred with it, using linear regression, viz.

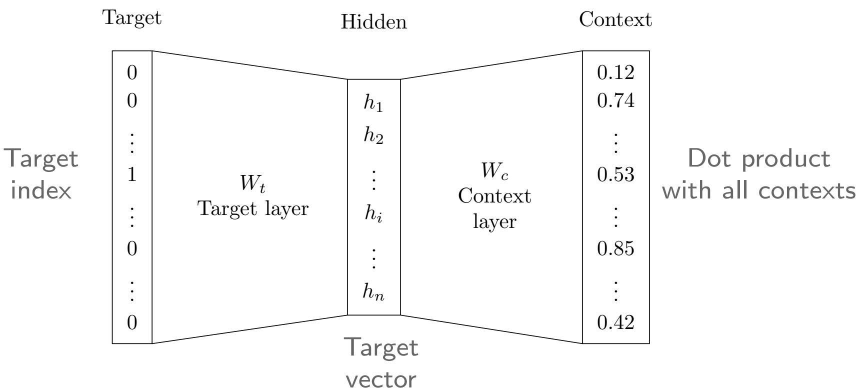
$$\forall \overrightarrow{An} : \overline{A} \cdot \overrightarrow{n} + \overrightarrow{b} \cong \overrightarrow{An}$$

Skipgram for nouns (1/2)

Goal Learn vectors for words such that two vectors have high similarity for co-occurring words, low similarity otherwise:

$$\cos(v_1, v_2) = \frac{v_1 \cdot v_2}{|v_1||v_2|}$$

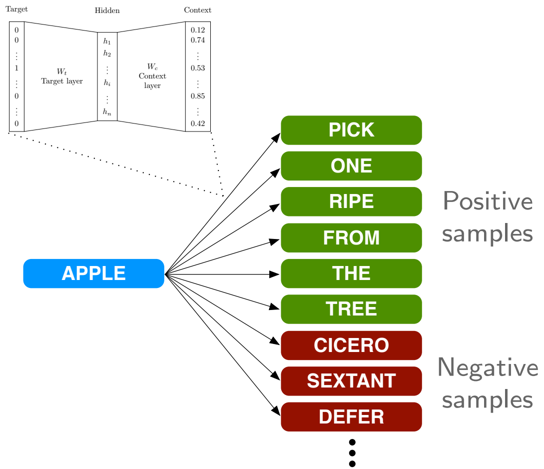
Network



Problem Need to compute dot product between all words in the vocabulary!

Skipgram for nouns (2/2)

Negative Sampling Instead of comparing against all contexts, we randomly sample contexts that are unlikely to appear for a given target word [Mikolov et al., 2013].

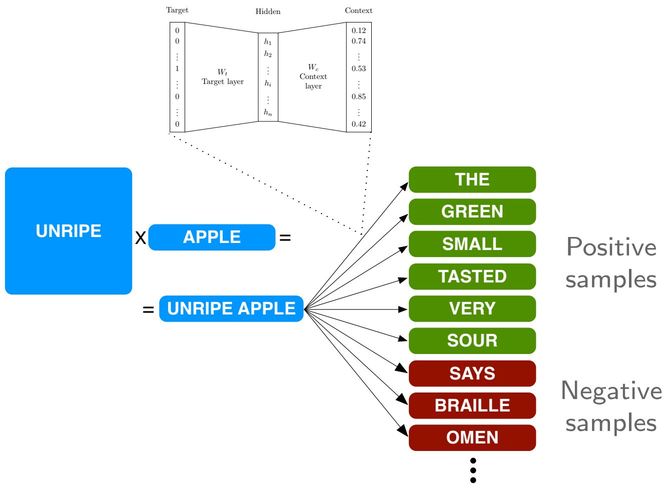


Formula

$$\sum_{c \in C} \log \sigma(\mathbf{n} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbf{n} \cdot \overline{\mathbf{c}})$$

Skipgram for adjectives

Context we swap the noun context for only the contexts for adjective-noun combinations, to learn a single matrix per adjective [Maillard and Clark, 2015]:



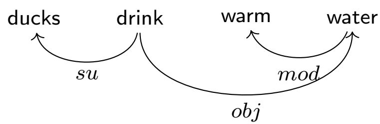
Formula

$$\sum_{c \in C} \log \sigma(\mathbf{An} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbf{An} \cdot \overline{\mathbf{c}})$$

Skipgram Tensors: in text

Intuïtion Let's try to generalize skipgram to arbitrary tensors. The syntactic type of a word determines the order of the tensor to be learnt; contexts are now tuples of words.

Example A transitive verb v with syntactic type $(np\backslash s)/np$, in vector space $\mathsf{N}\otimes\mathsf{S}\otimes\mathsf{N}$.



We obtain a word-context triple: $\{(\textbf{drink}, \text{duck}, \text{water})\}$

Puzzle.. We have to learn a cube, that combines with subject and object to get a sentence. But what is the context of a sentence?

Decomposition We decompose the cube into a pair of matrices: one transforms the subject to approximate the object, the other matrix vice versa [Wijnholds et al., 2020]:

$$\overline{\mathbf{drink}^s} \times \overrightarrow{duck} \cong \overrightarrow{water} \qquad \overline{\mathbf{drink}^o} \times \overrightarrow{water} \cong \overrightarrow{duck}$$

Skipgram Tensors: in formulas

For a word W with dependencies $d_1, d_2, ..., d_n$, we can define n different models by choosing how many dependents to use as context. This gives a trade off between tensor order and number of distinct tensors to learn:

Full tensor model

$$\sum_{c \in C} \log \sigma(\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_n \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_n \cdot \overline{\mathbf{c}})$$

N-1 model

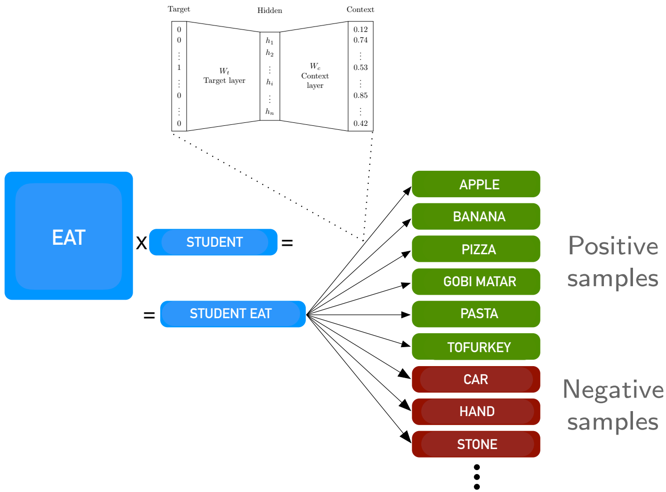
$$\sum_{d_i \in D} \log \sigma(\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_{i-1}\mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathbf{d}_i) + \sum_{\overline{d_i} \in \overline{D}} \log \sigma(-\mathbb{W}\mathbf{d}_1 ... \mathbf{d}_{i-1}\mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathbf{d}_i)$$

N-i model

$$\sum_{d_1 ... d_i \in D} \log \sigma(\mathbb{W}\mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathcal{P}_+\{\mathbf{d}_1, ..., \mathbf{d}_i\}) + \sum_{d_1 ... d_i \in D} \log \sigma(\mathbb{W}\mathbf{d}_{i+1} ... \mathbf{d}_n \cdot \mathcal{P}_+\{\mathbf{d}_1, ..., \mathbf{d}_i\})$$

Skipgram Tensors for verbs: N-1, subj model

Context objects that go with the verb subject combination under consideration:



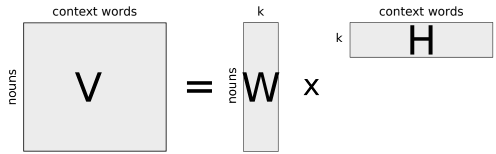
Formula

$$\sum_{o \in O} \log \sigma(\mathbf{V}\mathbf{s} \cdot \mathbf{o}) + \sum_{\overline{o} \in \overline{O}} \log \sigma(-\mathbf{V}\mathbf{s} \cdot \overline{\mathbf{o}})$$

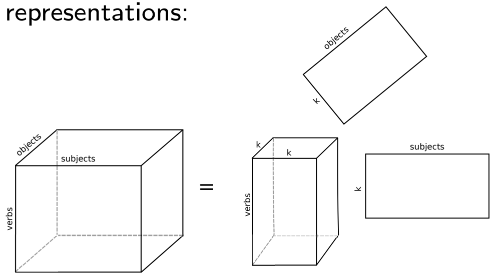
Tensor Factorization

Decomposition [Van de Cruys et al., 2013] proposes to use NMF and a variation on Tucker decomposition to approximate noun and verb representations:

1. Use non-negative matrix factorization (NMF) to retrieve dense noun vectors:

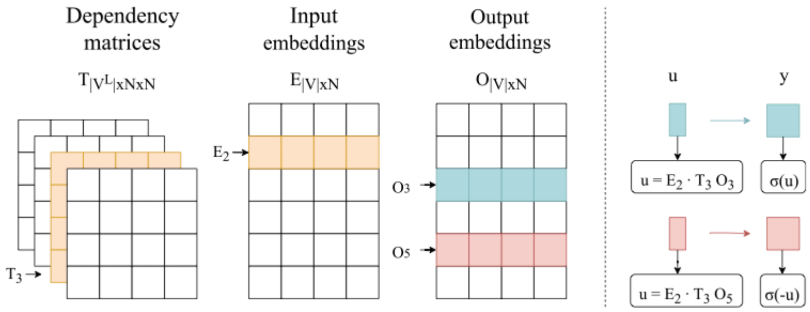


2. Collect (s, v, o) co-occurrence counts and factorize into a smaller tensor, against the noun vector representations:



Words are Vectors, Dependencies are Matrices

Dependencies as matrices As a scalable alternative, [Czarnowska et al., 2019] consider the skipgram model, where dependencies between a target word and a context word are represented by a separated matrix that transforms the context embedding:

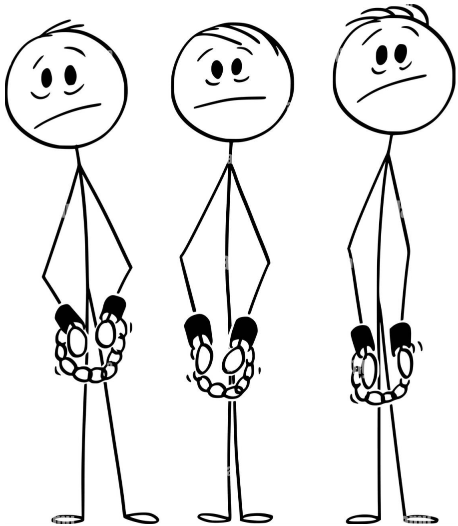


- Same context word and vector, different dependency, hence the dependency matrix transforms the context word to fit it's role in the sentence;
- In theory this approach could scale up, but it takes into account only dependencies, not syntactic types!

Discussion

- Different methods for learning representations have been explored
- Generally, taking the view that words ought to be higher-order tensors pending their syntactic type, leads to hard to scale methods or untractable learning.
- We need an approach that still takes syntactic/semantic types into account for modelling the relational behaviour of words, without sacrificing tractability.
- A more generalizable approach like the dependencies-as-matrices approach of [Czarnowska et al., 2019], still does not distinguish content from function words.

The Shackles of Linearity



Linear non-linearity

Trade offs We have traded in the formal semantics account of compositionality for a vector-based semantics that is strictly linear (algebraic)

Escapes As we can duplicate variables in lexical λ terms, we can replicate that process in a vector-based setting using Frobenius Algebras:

$$\lambda P \lambda Q \lambda x. P\ x \wedge Q\ x \qquad\qquad M_{ij} u_j\ \rho_{ikm}\ N_{kl} v_l$$

No escape? The example above is a *linear non-linearity*, as Frobenius Algebras are still linear in the (linear) algebraic sense.

Tractability The approaches to directly learn tensor representations, may use deep neural nets, but the representations are hard to learn and they end up being used in a linear algebraic way.

Deep Learning? Are we confined to the shackles of linearity? Or are there ways to escape the boundaries of linear algebra without compromizing compositionality?

Stay tuned...

References

Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In Proceedings of the 2010 conference on empirical methods in natural language processing, pages 1183–1193, 2010.

Paula Czarnowska, Guy Emerson, and Ann Copestake. Words are vectors, dependencies are matrices: Learning word embeddings from dependency graphs. 2019.

Jean Maillard and Stephen Clark. Learning adjective meanings with a tensor-based skip-gram model. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, pages 327–331, 2015.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, 2013.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. A tensor-based factorization model of semantic compositionality. In Conference of the North American Chapter of the Association of Computational Linguistics (HTL-NAACL), pages 1142–1151, 2013.

Gijs Wijnholds, Mehrnoosh Sadrzadeh, and Stephen Clark. Representation learning for type-driven composition. In Proceedings of the 24th Conference on Computational Natural Language Learning, pages 313–324, 2020.

Gijs Jasper Wijnholds. Coherent diagrammatic reasoning in compositional distributional semantics. In Logic, Language, Information, and Computation, pages 371–386. Springer Berlin Heidelberg, 2017. doi: 10.1007/978-3-662-55386-2_27. URL https://doi.org/10.1007%2F978-3-662-55386-2_27.