Learning High-Order Word Representations

Konstantinos Kogkalidis

June 12, 2018

LoLa Fan Club

Motivation

Distributional Compositional Semantics

Idea: structure-preserving map $\mathcal{F}$

$$\mathcal{F} : \mathcal{G} \to \textbf{\textit{FdVect}}$$

- Atomic types translated to vectors (order-one tensors)
- Complex types translated to (multi-)linear maps (higher order tensors)

Example

Word Type	$\mathcal{G}$ Type	$\mathcal{F}$ Translation
Noun	NP	$\mathbb{R}^{NP}$
Adjective	$NP \backslash NP$	$\mathbb{R}^{NP \times NP} \equiv \mathbb{R}^{NP} \to \mathbb{R}^{NP}$

$$\begin{aligned} \textbf{\textit{cat}} &\in \mathbb{R}^{NP} \\ \textbf{\textit{black}},\ \textbf{\textit{stray}} &\in \mathbb{R}^{NP \times NP} \\ \textbf{\textit{black stray cat}} &\in \mathbb{R}^{NP} \end{aligned}$$

Why Compositionality?

- Bridging of formal & distributional semantics
- Syntax-informed meaning derivations
- Modeling of functional words
- Formal treatment of ambiguous derivations
- Contextual Disambiguation
- Richer representations

  ⋮

Why Not Compositionality?

- ✓ Great properties
- ? How to obtain word representations?

Possible options:

Why Not Compositionality?

- ✓ Great properties
- **?** How to obtain word representations?

Possible options:

1. Co-occurrence statistics

Why Not Compositionality?

- ✓ Great properties
- **?** How to obtain word representations?

Possible options:

1. Co-occurrence statistics ✗
2. Unsupervised techniques (*a la word2vec*)

Why Not Compositionality?

- ✓ Great properties
- **?** How to obtain word representations?

Possible options:

1. Co-occurrence statistics ✗
2. Unsupervised techniques (*a la word2vec*) ✗
3. Supervised learning **?**

Problem Statement

Examine whether supervised learning can be used to find higher-order word representations (transitive verbs)

Supervised Learning

Overview

- Search over set of functions $A \to B$ parameterized over P
- Find optimal approximation $\hat{f}_P$ to $f : A \to B$
- Use samples $(a, f(a)) \in A \times B$ to update P

Supervised Learning

Dataset

Finding Data

Sample space must be:

- labeled
- constrained
- of large size
- of high quality

Paraphrase Database

Raw text paraphrase pairs

Example pair

proposed by the president $\sim$ *suggested by the chairman*

- labeled ✓
- constrained ✗ *(different syntactic types)*
- of large size ✓
- of high quality **?**

Dataset: Preprocessing

1. **Parse and filter by type** (transitive verb case)

- labeled ✓
- constrained ✓
- of large size ✗ *(>95% loss)*
- of high quality ✗ *(parser-induced errors)*

Dataset: Preprocessing

2. **Back-translation**

- Labeled ✓
- constrained ✓
- of large size ✓
- of high quality ✗ *(translation-induced errors)*

Dataset: Preprocessing

3. **Filter by co-occurrence / mutual information**

- Labeled ✓
- constrained ✓
- of large size ✓
- of high quality **?**

Dataset: End Result

Verb / object dictionaries:

$$\mathcal{V} : \{v_1 : 1, v_2 : 2, ..., v_N : N\}$$
$$\mathcal{O} : \{o_1 : 1, o_2 : 2, ..., o_M : M\}$$

Paraphrase relation:

$$\mathcal{P} : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \{0, 1\} \qquad \text{(binary classification)}$$
$$\mathcal{P}(i, j, k, l) = \mathcal{P}(k, l, i, j) = \begin{cases} 1 & v_i o_j \sim v_k o_l \\ 0 & \text{otherwise} \end{cases}$$

Supervised Learning

Formulating the Network

Training Objective

Our semantic interpretations are:

- Actions: $\lceil a \rceil = \mathbb{R}^A$
- Objects: $\lceil np \rceil = \mathbb{R}^{NP}$
- Transitive Verbs: $\lceil a/np \rceil = \mathbb{R}^{A \times NP}$

And our objective is to learn a verb embedding function ε_{verb}:

$$\varepsilon_{verb} : \mathbb{N} \to \mathbb{R}^{A \times NP}$$

But instead we have samples from some $f : \mathbb{N}^4 \to \{0, 1\}$

Intermediate Representations

Solution

Formulate f_P to incorporate ε_{verb}.

$$f_p = f_1 \circ f_2 \circ \cdots \circ \varepsilon_{verb} \circ \ldots$$

Intermediate Representations

Solution

Formulate f_P to incorporate ε_{verb}.

$$f_p = f_1 \circ f_2 \circ \cdots \circ \varepsilon_{verb} \circ \ldots$$

Simplification (1)

Assume pre-trained object embedding function ε_{object}.

$$\varepsilon_{object} : \mathbb{N} \rightarrow \mathbb{R}^{300}$$

Filling the missing blocks

$i \in \mathbb{N}$ $j \in \mathbb{N}$ $k \in \mathbb{N}$ $l \in \mathbb{N}$

$\hat{y} \in \mathbb{R}$

Filling the missing blocks

$$
\begin{tikzcd}
i \in \mathbb{N} & j \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] & k \in \mathbb{N} & l \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] \\
 & \boldsymbol{j} \in \mathbb{R}^{300} & & \boldsymbol{l} \in \mathbb{R}^{300}
\end{tikzcd}
$$

$$\hat{y} \in \mathbb{R}$$

Filling the missing blocks

$$\begin{array}{cccc}
i \in \mathbb{N} & j \in \mathbb{N} & k \in \mathbb{N} & l \in \mathbb{N} \\
\downarrow \varepsilon_{verb} & \downarrow \varepsilon_{object} & \downarrow \varepsilon_{verb} & \downarrow \varepsilon_{object} \\
\boldsymbol{\mathcal{I}} \in \mathbb{R}^{100\times 300} & \boldsymbol{j} \in \mathbb{R}^{300} & \boldsymbol{\mathcal{K}} \in \mathbb{R}^{100\times 300} & \boldsymbol{l} \in \mathbb{R}^{300}
\end{array}$$

$$\hat{y} \in \mathbb{R}$$

Filling the missing blocks

$$
\begin{tikzcd}
i \in \mathbb{N} \arrow[d, "\varepsilon_{verb}"] & j \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] & k \in \mathbb{N} \arrow[d, "\varepsilon_{verb}"] & l \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] \\
\boldsymbol{\mathcal{I}} \in \mathbb{R}^{100\times 300} \arrow[r, bend left, no head] \arrow[dr, bend right] & \boldsymbol{j} \in \mathbb{R}^{300} & \boldsymbol{\mathcal{K}} \in \mathbb{R}^{100\times 300} \arrow[r, bend left, no head] \arrow[dr, bend right] & \boldsymbol{l} \in \mathbb{R}^{300} \\
 & \boldsymbol{\mathcal{I}}(\boldsymbol{j}) \in \mathbb{R}^{100} & & \boldsymbol{\mathcal{K}}(\boldsymbol{l}) \in \mathbb{R}^{100}
\end{tikzcd}
$$

$$\hat{y} \in \mathbb{R}$$

Filling the missing blocks

$$
\begin{tikzcd}
i \in \mathbb{N} \arrow[d, "\varepsilon_{verb}"] & j \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] & k \in \mathbb{N} \arrow[d, "\varepsilon_{verb}"] & l \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] \\
\mathcal{I} \in \mathbb{R}^{100\times 300} \arrow[dr] & \boldsymbol{j} \in \mathbb{R}^{300} & \mathcal{K} \in \mathbb{R}^{100\times 300} \arrow[dr] & \boldsymbol{l} \in \mathbb{R}^{300} \\
& \mathcal{I}(\boldsymbol{j}) \in \mathbb{R}^{100} \arrow[dr, "cosine"] & & \mathcal{K}(\boldsymbol{l}) \in \mathbb{R}^{100} \arrow[dl] \\
& & \hat{y} \in \mathbb{R} &
\end{tikzcd}
$$

Sanity Check

Objective Function

$$cos(\mathbf{V}_i(\mathbf{j}), \mathbf{V}_k(\mathbf{l})) \rightsquigarrow \mathcal{P}(i, j, k, l) \quad \forall \, (i, j, k, l) \in \mathcal{V} \times \mathcal{O} \times \mathcal{V} \times \mathcal{O}$$

Sanity Check

Objective Function

$$cos(\boldsymbol{V}_i(\boldsymbol{j}), \boldsymbol{V}_k(\boldsymbol{l})) \rightsquigarrow \mathcal{P}(i, j, k, l) \quad \forall\ (i, j, k, l) \in \mathcal{V} \times \mathcal{O} \times \mathcal{V} \times \mathcal{O}$$

Considerations:

1. Network Size
    - 1.000 verbs
    - $100 \times 300 = 30.000$ parameters per verb

    $\Rightarrow$ 30.000.000 parameters for ε_{verb} to learn

Sanity Check

Objective Function

$$cos(\boldsymbol{V}_i(\boldsymbol{j}), \boldsymbol{V}_k(\boldsymbol{l})) \rightsquigarrow \mathcal{P}(i,j,k,l) \quad \forall\, (i,j,k,l) \in \mathcal{V} \times \mathcal{O} \times \mathcal{V} \times \mathcal{O}$$

Considerations:

1. Network Size
    - 1.000 verbs
    - $100 \times 300 = 30.000$ parameters per verb

   $\Rightarrow$ 30.000.000 parameters for ε_{verb} to learn
2. Quantifying over two spaces ...

Sanity Check

Objective Function

$$\cos(\mathbf{V}_i(\mathbf{j}), \mathbf{V}_k(\mathbf{l})) \rightsquigarrow \mathcal{P}(i, j, k, l) \quad \forall\, (i, j, k, l) \in \mathcal{V} \times \mathcal{O} \times \mathcal{V} \times \mathcal{O}$$

Considerations:

1. Network Size
    - 1.000 verbs
    - $100 \times 300 = 30.000$ parameters per verb

    $\Rightarrow$ 30.000.000 parameters for ε_{verb} to learn
2. Quantifying over two spaces ...
3. ... both of which are non-convex

… A ”beast” to train ☹

Supervised Learning

Transferring Knowledge

Finding an Oracle

Dataception: use our labeled dataset to create a new dataset

Finding an Oracle

Dataception: use our labeled dataset to create a new dataset

Simplification (2)

Assume another pre-trained verb embedding function ε'_{verb}.

$$\varepsilon'_{verb} : \mathbb{N} \to \mathbb{R}^{300}$$

Finding an Oracle

Dataception: use our labeled dataset to create a new dataset

Simplification (2)

Assume another pre-trained verb embedding function ε'_{verb}.

$$\varepsilon'_{verb} : \mathbb{N} \to \mathbb{R}^{300}$$

Oracle

We can now train a paraphrase embedding function ε_{par}.

$$\varepsilon_{par} : \mathbb{R}^{300} \times \mathbb{R}^{300} \to \mathbb{R}^{100}$$

Oracle Flow

$i \in \mathbb{N}$ $j \in \mathbb{N}$ $k \in \mathbb{N}$ $l \in \mathbb{N}$

$\hat{y} \in \mathbb{R}$

Oracle Flow

$$
\begin{array}{cccc}
i \in \mathbb{N} & j \in \mathbb{N} & k \in \mathbb{N} & l \in \mathbb{N} \\
 & \downarrow \varepsilon_{object} & & \downarrow \varepsilon_{object} \\
 & \boldsymbol{j} \in \mathbb{R}^{300} & & \boldsymbol{l} \in \mathbb{R}^{300}
\end{array}
$$

$$\hat{y} \in \mathbb{R}$$

Oracle Flow

$$
\begin{tikzcd}
i \in \mathbb{N} \arrow[d, "\varepsilon'_{verb}"] & j \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] & k \in \mathbb{N} \arrow[d, "\varepsilon'_{verb}"] & l \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] \\
\boldsymbol{i} \in \mathbb{R}^{300} & \boldsymbol{j} \in \mathbb{R}^{300} & \boldsymbol{k} \in \mathbb{R}^{300} & \boldsymbol{l} \in \mathbb{R}^{300}
\end{tikzcd}
$$

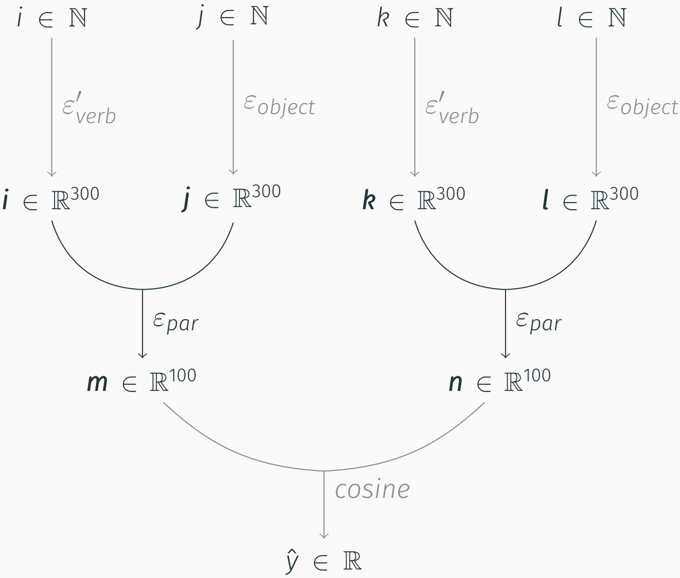
$$\hat{y} \in \mathbb{R}$$

Oracle Flow

$$
\begin{tikzcd}
i \in \mathbb{N} \arrow[d, "\varepsilon'_{verb}"] & & j \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] & k \in \mathbb{N} \arrow[d, "\varepsilon'_{verb}"] & & l \in \mathbb{N} \arrow[d, "\varepsilon_{object}"] \\
\boldsymbol{i} \in \mathbb{R}^{300} \arrow[dr, no head] & & \boldsymbol{j} \in \mathbb{R}^{300} \arrow[dl, "\varepsilon_{par}"] & \boldsymbol{k} \in \mathbb{R}^{300} \arrow[dr, no head] & & \boldsymbol{l} \in \mathbb{R}^{300} \arrow[dl, "\varepsilon_{par}"] \\
& \boldsymbol{m} \in \mathbb{R}^{100} & & & \boldsymbol{n} \in \mathbb{R}^{100} &
\end{tikzcd}
$$

$$\hat{y} \in \mathbb{R}$$

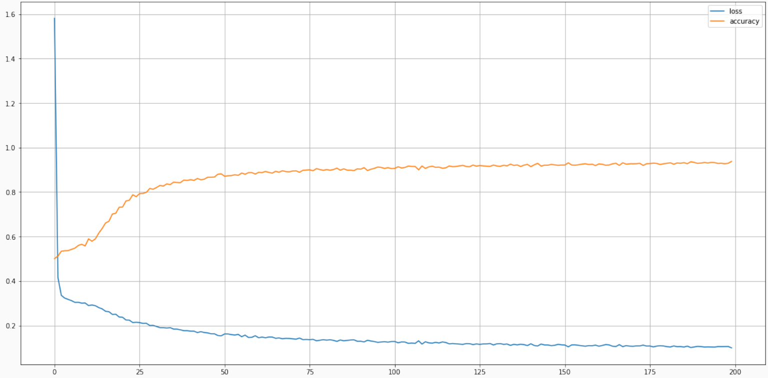
Oracle Flow



$i \in \mathbb{N}$
$j \in \mathbb{N}$
$k \in \mathbb{N}$
$l \in \mathbb{N}$
ε'_{verb}
ε_{object}
ε'_{verb}
ε_{object}
$\boldsymbol{i} \in \mathbb{R}^{300}$
$\boldsymbol{j} \in \mathbb{R}^{300}$
$\boldsymbol{k} \in \mathbb{R}^{300}$
$\boldsymbol{l} \in \mathbb{R}^{300}$
ε_{par}
ε_{par}
$\boldsymbol{m} \in \mathbb{R}^{100}$
$\boldsymbol{n} \in \mathbb{R}^{100}$
cosine
$\hat{y} \in \mathbb{R}$

Training the Oracle

ε_{par}: recurrent autoencoder ($\approx$ 700.000 parameters)



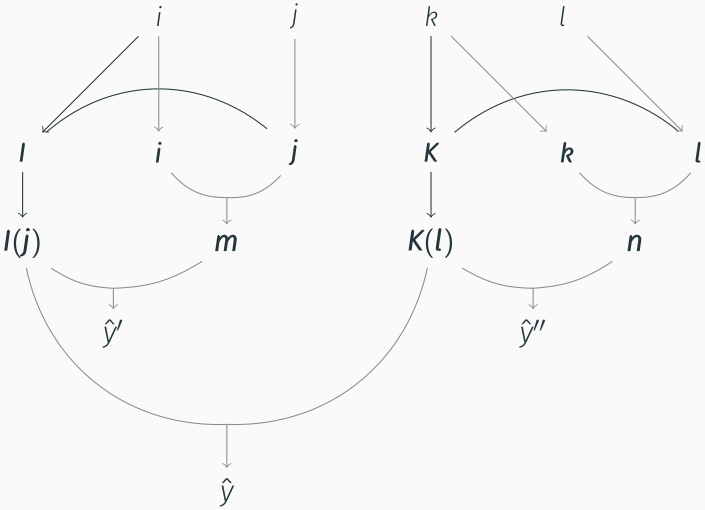
Utilizing the Oracle

New Objective Function

$$cos(\boldsymbol{V}_i(\boldsymbol{j}), \varepsilon_{par}(\boldsymbol{v}, \boldsymbol{j})) \rightsquigarrow 1 \quad \forall\ (i, j) \in \mathcal{V} \times \mathcal{O}$$

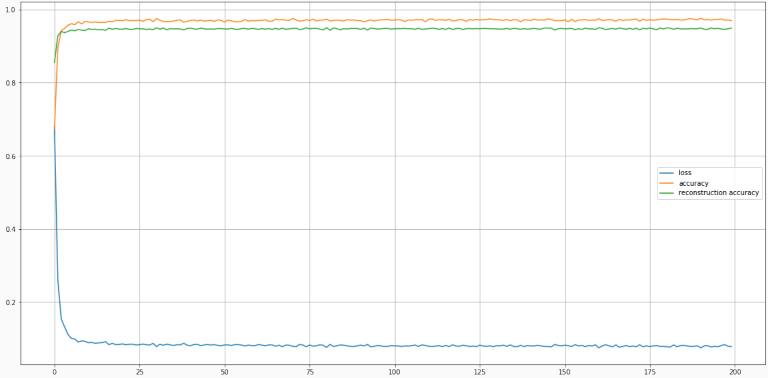
- ε_{par} gives us paraphrase embeddings '*for free*'
- We can use them to facilitate training
- Much smaller problem space

Composing Networks



Training the Original

ε_{verb}: tanh activated dense layer ($\approx$ 30.000.000 parameters)



The beast has been tamed! ☺

Evaluation

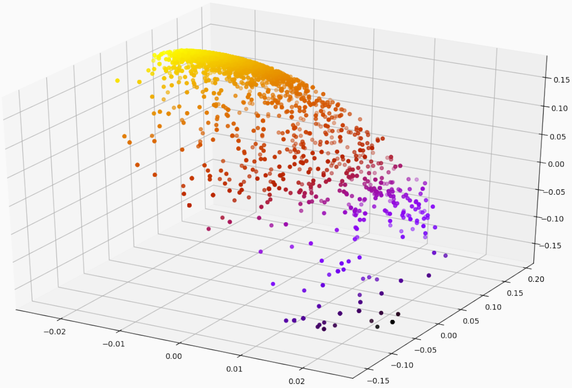
Microstructure

Task-specific performance relates to the small-scale structure of the learned space:

Ground Truth \ Prediction	Oracle		Final	
	$\top$	$\bot$	$\top$	$\bot$
$\top$	0.92	0.08	0.88	0.02
$\bot$	0.08	0.92	0.12	0.98

Macrostructure

3D PCA on paraphrase embeddings



Conclusion

Critique

1. Uninformative error signal **?**
2. Over-parameterization
    a) Encoder / decoder architectures **?**
    b) Linearity constraint ✗
    c) Chasing after an oracle **?**
    d) Bad scaling ✗

Next Steps

1. Directly evaluate verb matrices
2. More structural constraints (activity regularization)
3. Iterative learning
4. Other data formats:
    - Different syntactic types
    - Different labels / samples altogether
5. Different oracle architectures

Bag of Tricks

1. Negative Sampling
2. Curriculum Learning / Domain Adaptation
3. Loss Mixing