When the Whole Is Not Greater Than the Combination of Its Parts: A "Decompositional" Look at Compositional Distributional Semantics

Fabio Massimo Zanzotto*
University of Rome "Tor Vergata"

Lorenzo Ferrone
University of Rome "Tor Vergata"

Marco Baroni
University of Trento

Distributional semantics has been extended to phrases and sentences by means of composition operations. We look at how these operations affect similarity measurements, showing that similarity equations of an important class of composition methods can be decomposed into operations performed on the subparts of the input phrases. This establishes a strong link between these models and convolution kernels.

1. Introduction

Distributional semantics approximates word meanings with vectors tracking co-occurrence in corpora (Turney and Pantel 2010). Recent work has extended this approach to phrases and sentences through vector composition (Clark 2015). Resulting **compositional distributional semantic models** (CDSMs) estimate degrees of semantic similarity (or, more generally, relatedness) between two phrases: A good CDSM might tell us that *green bird* is closer to *parrot* than to *pigeon*, useful for tasks such as paraphrasing.

We take a mathematical look[1] at how the composition operations postulated by CDSMs affect similarity measurements involving the vectors they produce for phrases or sentences. We show that, for an important class of composition methods, encompassing at least those based on linear transformations, the similarity equations can be decomposed into operations performed on the subparts of the input phrases,

* Department of Enterprise Engineering, University of Rome "Tor Vergata," Viale del Politecnico, 1, 00133 Rome, Italy. E-mail: fabio.massimo.zanzotto@uniroma2.it.

1 Ganesalingam and Herbelot (2013) also present a mathematical investigation of CDSMs. However, except for the tensor product (a composition method we do not consider here as it is not empirically effective), they do not look at how composition strategies affect similarity comparisons.

Original submission received: 10 December 2013; revision received: 26 May 2014; accepted for publication: 1 August 2014.

doi:10.1162/COLI_a_00215

Table 1
Compositional Distributional Semantic Models: $\vec{a}$, $\vec{b}$, and $\vec{c}$ are distributional vectors representing the words a, b, and c, respectively; matrices $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are constant across a phrase type, corresponding to syntactic slots; the matrix $\mathbf{A}$ and the third-order tensor $\mathbb{B}$ represent the predicate words a in the first phrase and b in the second phrase, respectively.

	2-word phrase	3-word phrase	reference
Additive	$\vec{a}+\vec{b}$	$\vec{a}+\vec{b}+\vec{c}$	Mitchell and Lapata (2008)
Multiplicative	$\vec{a}\boxdot\vec{b}$	$\vec{a}\boxdot\vec{b}\boxdot\vec{c}$	Mitchell and Lapata (2008)
Full Additive	$\mathbf{X}\vec{a}+\mathbf{Y}\vec{b}$	$\mathbf{Y}\vec{a}+\mathbf{X}\vec{b}+\mathbf{Z}\vec{c}$	Guevara (2010), Zanzotto et al. (2010)
Lexical Function	$\mathbf{A}\vec{b}$	$\vec{a}\,\mathbb{B}\,\vec{c}$	Coecke, Sadrzadeh, and Clark (2010)

and typically factorized into terms that reflect the linguistic structure of the input. This establishes a strong link between CDSMs and convolution kernels (Haussler 1999), which act in the same way. We thus refer to our claim as the "Convolution Conjecture."

We focus on the models in Table 1. These CDSMs all apply linear methods, and we suspect that linearity is a sufficient (but not necessary) condition to ensure that the Convolution Conjecture holds. We will first illustrate the conjecture for linear methods, and then briefly consider two nonlinear approaches: the **dual space** model of Turney (2012), for which it does, and a representative of the recent strand of work on neural-network models of composition, for which it does not.

2. Mathematical Preliminaries

Vectors are represented as small letters with an arrow $\vec{a}$ and their elements are a_i, matrices as capital letters in bold $\mathbf{A}$ and their elements are A_{ij}, and third-order or fourth-order tensors as capital letters in the form $\mathbb{A}$ and their elements are A_{ijk} or A_{ijkh}. The symbol $\boxdot$ represents the element-wise product and $\otimes$ is the tensor product. The dot product is $\langle\vec{a},\vec{b}\rangle$ and the Frobenius product—that is, the generalization of the dot product to matrices and high-order tensors—is represented as $\langle\mathbf{A},\mathbf{B}\rangle_F$ and $\langle\mathbb{A},\mathbb{B}\rangle_F$. The Frobenius product acts on vectors, matrices, and third-order tensors as follows:

$$\langle\vec{a},\vec{b}\rangle_F=\sum_i a_i b_i=\langle\vec{a},\vec{b}\rangle \qquad \langle\mathbf{A},\mathbf{B}\rangle_F=\textstyle\sum_{ij}A_{ij}B_{ij} \qquad \langle\mathbb{A},\mathbb{B}\rangle_F=\sum_{ijk}A_{ijk}B_{ijk} \tag{1}$$

A simple property that relates the dot product between two vectors and the Frobenius product between two general tensors is the following:

$$\langle\vec{a},\vec{b}\rangle=\langle\mathbf{I},\vec{a}\vec{b}^T\rangle_F \tag{2}$$

where $\mathbf{I}$ is the identity matrix. The dot product of $\mathbf{A}\vec{x}$ and $\mathbf{B}\vec{y}$ can be rewritten as:

$$\langle\mathbf{A}\vec{x},\mathbf{B}\vec{y}\rangle=\langle\mathbf{A}^T\mathbf{B},\vec{x}\vec{y}^T\rangle_F \tag{3}$$

Let $\mathbf{A}$ and $\mathbf{B}$ be two third-order tensors and $\vec{x}, \vec{y}, \vec{a}, \vec{c}$ four vectors. It can be shown that:

$$\langle \vec{x}\mathbf{A}\vec{y}, \vec{a}\mathbf{B}\vec{c}\rangle = \langle \sum_j (\mathbf{A} \otimes \mathbf{B})_j, \vec{x} \otimes \vec{y} \otimes \vec{a} \otimes \vec{c}\rangle_F \tag{4}$$

where $\mathbf{C} = \sum_j (\mathbf{A} \otimes \mathbf{B})_j$ is a non-standard way to indicate the tensor contraction of the tensor product between two third-order tensors. In this particular tensor contraction, the elements C_{iknm} of the resulting fourth-order tensor $\mathbf{C}$ are $C_{iknm} = \sum_j A_{ijk}B_{njm}$. The elements D_{iknm} of the tensor $\mathbf{D} = \vec{x} \otimes \vec{y} \otimes \vec{a} \otimes \vec{c}$ are $D_{iknm} = x_i y_k a_n c_m$.

3. Formalizing the Convolution Conjecture

Structured Objects. In line with Haussler (1999), a **structured object** $x \in X$ is either a terminal object that cannot be furthermore decomposed, or a non-terminal object that can be decomposed into n subparts. We indicate with $\overline{\mathbf{x}} = (x_1, \ldots, x_n)$ one such decomposition, where the subparts $x_i \in X$ are structured objects themselves. The set X is the set of the structured objects and $T_X \subseteq X$ is the set of the terminal objects. A structured object x can be anything according to the representational needs. Here, x is a representation of a text fragment, and so it can be a sequence of words, a sequence of words along with their part of speech, a tree structure, and so on. The set $\mathcal{R}(x)$ is the set of decompositions of x relevant to define a specific CDSM. Note that a given decomposition of a structured object x does not need to contain all the subparts of the original object. For example, let us consider the phrase $x =$ *tall boy*. We can then define $\mathcal{R}(x) = \{(\text{tall}, \text{boy}), (\text{tall}), (\text{boy})\}$. This set contains the three possible decompositions of the phrase: $(\underbrace{\text{tall}}_{x_1}, \underbrace{\text{boy}}_{x_2})$, $(\underbrace{\text{tall}}_{x_1})$, and $(\underbrace{\text{boy}}_{x_1})$.

Recursive formulation of CDSM. A CDSM can be viewed as a function f that acts recursively on a structured object x. If x is a non-terminal object

$$f(x) = \bigodot_{\overline{\mathbf{x}} \in \mathcal{R}(x)} \gamma(f(x_1), f(x_2), \ldots, f(x_n)) \tag{5}$$

where $\mathcal{R}(x)$ is the set of relevant decompositions, $\bigodot$ is a repeated operation on this set, γ is a function defined on $f(x_i)$ where x_i are the subparts of a decomposition of x. If x is a terminal object, $f(x)$ is directly mapped to a tensor. The function f may operate differently on different kinds of structured objects, with tensor degree varying accordingly. The set $\mathcal{R}(x)$ and the functions f, γ, and $\bigodot$ depend on the specific CDSM, and the same CDSM might be susceptible to alternative analyses satisfying the form in Equation (5). As an example, under Additive, x is a sequence of words and f is

$$f(x) = \begin{cases} \sum_{y \in \mathcal{R}(x)} f(y) & \text{if } x \notin T_X \\ \vec{x} & \text{if } x \in T_X \end{cases} \tag{6}$$

where $\mathcal{R}((w_1, \ldots, w_n)) = \{(w_1), \ldots, (w_n)\}$. The repeated operation $\bigodot$ corresponds to summing and γ is identity. For Multiplicative we have

$$f(x) = \begin{cases} \boxdot_{y \in \mathcal{R}(x)} f(y) & \text{if } x \notin T_X \\ \vec{x} & \text{if } x \in T_X \end{cases} \tag{7}$$

where $\mathcal{R}(x) = \{(w_1, \ldots, w_n)\}$ (a single trivial decomposition including all subparts). With a single decomposition, the repeated operation reduces to a single term; and here γ is the product (it will be clear subsequently, when we apply the Convolution Conjecture to these models, why we are assuming different decomposition sets for Additive and Multiplicative).

Definition 1 (Convolution Conjecture)
For every CDSM f along with its $\mathcal{R}(x)$ set, there exist functions K, K_i and a function g such that:

$$K(f(x), f(y)) = \sum_{\substack{\overline{\mathbf{x}} \in \mathcal{R}(x) \\ \overline{\mathbf{y}} \in \mathcal{R}(y)}} g(K_1(f(x_1), f(y_1)), K_2(f(x_2), f(y_2)), \ldots, K_n(f(x_n), f(y_n))) \tag{8}$$

The Convolution Conjecture postulates that the similarity $K(f(x), f(y))$ between the tensors $f(x)$ and $f(y)$ is computed by combining operations on the subparts, that is, $K_i(f(x_i), f(y_i))$, using the function g. This is exactly what happens in convolution kernels (Haussler 1999). K is usually the dot product, but this is not necessary: We will show that for the dual-space model of Turney (2012) K turns out to be the fourth root of the Frobenius tensor.

4. Comparing Composed Phrases

We illustrate now how the Convolution Conjecture (CC) applies to the considered CDSMs, exemplifying with adjective–noun and subject–verb–object phrases. Without loss of generality we use *tall boy* and *red cat* for adjective–noun phrases and *goats eat grass* and *cows drink water* for subject–verb–object phrases.

Additive Model. K and K_i are dot products, g is the identity function, and f is as in Equation (6). The structure of the input is a word sequence (i.e., $x = (w_1\, w_2)$) and the relevant decompositions consist of these single words, $\mathcal{R}(x) = \{(w_1), (w_2)\}$. Then

$$\begin{aligned} K(f(\text{tall boy}), f(\text{red cat})) &= \langle \vec{tall} + \vec{boy}, \vec{red} + \vec{cats} \rangle = \\ &= \langle \vec{tall}, \vec{red} \rangle + \langle \vec{tall}, \vec{cat} \rangle + \langle \vec{boy}, \vec{red} \rangle + \langle \vec{boy}, \vec{cat} \rangle = \\ &= \sum_{\substack{x \in \{\text{tall,boy}\} \\ y \in \{\text{red,cat}\}}} \langle f(x), f(y) \rangle = \sum_{\substack{x \in \{\text{tall,boy}\} \\ y \in \{\text{red,cat}\}}} K(f(x), f(y)) \end{aligned} \tag{9}$$

The CC form of Additive shows that the overall dot product can be decomposed into dot products of the vectors of the single words. Composition does not add any further information. These results can be easily extended to longer phrases and to phrases of different length.

Multiplicative Model. K, g are dot products, K_i the component-wise product, and f is as in Equation (7). The structure of the input is $x = (w_1\, w_2)$, and we use the trivial single decomposition consisting of all subparts (thus summation reduces to a single term):

$$\begin{aligned} K(f(\text{tall boy}), f(\text{red cat})) &= \langle \vec{tall} \boxdot \vec{boy}, \vec{red} \boxdot \vec{cat} \rangle = \langle \vec{tall} \boxdot \vec{red} \boxdot \vec{boy} \boxdot \vec{cat}, \vec{1} \rangle = \\ &= \langle \vec{tall} \boxdot \vec{red}, \vec{boy} \boxdot \vec{cat} \rangle = g(K_1(\vec{tall}, \vec{red}), K_2(\vec{boy}, \vec{cat})) \end{aligned} \tag{10}$$

This is the dot product between an indistinct chain of element-wise products and a vector $\vec{1}$ of all ones or the product of two separate element-wise products, one on adjectives $\vec{tall} \boxdot \vec{red}$, and one on nouns $\vec{boy} \boxdot \vec{cat}$. In this latter CC form, the final dot product is obtained in two steps: first separately operating on the adjectives and on the nouns; then taking the dot product of the resulting vectors. The comparison operations are thus reflecting the input syntactic structure. The results can be easily extended to longer phrases and to phrases of different lengths.

Full Additive Model. The input consists of a sequence of (label,word) pairs $x = ((L_1\, w_1), \ldots, (L_n\, w_n))$ and the relevant decomposition set includes the single tuples, that is, $\mathcal{R}(x) = \{(L_1\, w_1), \ldots, (L_n\, w_n)\}$. The CDSM f is defined as

$$f(x) = \begin{cases} \sum_{(L\, w) \in \mathcal{R}(x)} f(L) f(w) & \text{if } x \notin T_X \\ \mathbf{X} & \text{if } x \in T_X \text{ is a label } L \\ \vec{w} & \text{if } x \in T_X \text{ is a word } w \end{cases} \tag{11}$$

The repeated operation $\bigodot$ here is summation, and γ the matrix-by-vector product. In the CC form, K is the dot product, g the Frobenius product, $K_1(f(x), f(y)) = f(x)^T f(y)$, and $K_2(f(x), f(y)) = f(x) f(y)^T$. We have then for adjective–noun composition (by using the property in Equation (3)):

$$\begin{aligned} & K(f((\text{A tall})\ (\text{N boy})), f((\text{A red})\ (\text{N cat}))) = \langle \mathbf{A}\vec{tall} + \mathbf{N}\vec{boy}, \mathbf{A}\vec{red} + \mathbf{N}\vec{cat} \rangle = \\ & = \langle \mathbf{A}\vec{tall}, \mathbf{A}\vec{red} \rangle + \langle \mathbf{A}\vec{tall}, \mathbf{N}\vec{cat} \rangle + \langle \mathbf{N}\vec{boy}, \mathbf{A}\vec{red} \rangle + \langle \mathbf{N}\vec{boy}, \mathbf{N}\vec{cat} \rangle = \\ & = \langle \mathbf{A}^T\mathbf{A}, \vec{tall}\,\vec{red}^T \rangle_F + \langle \mathbf{N}^T\mathbf{A}, \vec{boy}\,\vec{red}^T \rangle_F + \langle \mathbf{A}^T\mathbf{N}, \vec{tall}\,\vec{cat}^T \rangle_F + \langle \mathbf{N}^T\mathbf{N}, \vec{boy}\,\vec{cat}^T \rangle_F = \\ & \quad = \sum_{\substack{(l_x\, w_x) \in \{(\text{A tall}), (\text{N boy})\} \\ (l_y\, w_y) \in \{(\text{A red}), (\text{N cat})\}}} g(K_1(f(l_x), f(l_y)), K_2(f(w_x), f(w_y)) \end{aligned} \tag{12}$$

The CC form shows how Full Additive factorizes into a more structural and a more lexical part: Each element of the sum is the Frobenius product between the product of two matrices representing syntactic labels and the tensor product between two vectors representing the corresponding words. For subject–verb–object phrases $((S\,w_1)\,(V\,w_2)\,(O\,w_3))$ we have

$$\begin{aligned}
&K(f(((S\,goats)\,(V\,eat)\,(O\,grass))), f(((S\,cows)\,(V\,drink)\,(O\,water)))) = \\
&= \langle \mathbf{S}\,\vec{goats} + \mathbf{V}\,\vec{eat} + \mathbf{O}\,\vec{grass}, \mathbf{S}\,\vec{cows} + \mathbf{V}\,\vec{drink} + \mathbf{O}\,\vec{water} \rangle = \\
&= \langle \mathbf{S}^T\mathbf{S}, \vec{goats}\,\vec{cows}^T \rangle_F + \langle \mathbf{S}^T\mathbf{V}, \vec{goats}\,\vec{drink}^T \rangle_F + \langle \mathbf{S}^T\mathbf{O}, \vec{goats}\,\vec{water}^T \rangle_F \\
&+ \langle \mathbf{V}^T\mathbf{S}, \vec{eat}\,\vec{cows}^T \rangle_F + \langle \mathbf{V}^T\mathbf{V}, \vec{eat}\,\vec{drink}^T \rangle_F + \langle \mathbf{V}^T\mathbf{O}, \vec{eat}\,\vec{water}^T \rangle_F \\
&+ \langle \mathbf{O}^T\mathbf{S}, \vec{grass}\,\vec{cows}^T \rangle_F + \langle \mathbf{O}^T\mathbf{V}, \vec{grass}\,\vec{drink}^T \rangle_F + \langle \mathbf{O}^T\mathbf{O}, \vec{grass}\,\vec{water}^T \rangle_F \\
&= \sum_{\substack{(l_x\,w_x) \in \{(S\,goats),(V\,eat),(O\,grass)\} \\ (l_y\,w_y) \in \{(S\,cows),(V\,drink),(O\,water)\}}} g(K_1(f(l_x), f(l_y)), K_2(f(w_x), f(w_y))
\end{aligned} \tag{13}$$

Again, we observe the factoring into products of syntactic and lexical representations.

By looking at Full Additive in the CC form, we observe that when $\mathbf{X}^T\mathbf{Y} \approx \mathbf{I}$ for all matrix pairs, it degenerates to Additive. Interestingly, Full Additive can also approximate a *semantic* convolution kernel (Mehdad, Moschitti, and Zanzotto 2010), which combines dot products of elements in the same slot. In the adjective–noun case, we obtain this approximation by choosing two nearly orthonormal matrices $\mathbf{A}$ and $\mathbf{N}$ such that $\mathbf{A}\mathbf{A}^T = \mathbf{N}\mathbf{N}^T \approx \mathbf{I}$ and $\mathbf{A}\mathbf{N}^T \approx \mathbf{0}$ and applying Equation (2): $\langle \mathbf{A}\,\vec{tall} + \mathbf{N}\,\vec{boy}, \mathbf{A}\,\vec{red} + \mathbf{N}\,\vec{cat} \rangle \approx \langle tall, red \rangle + \langle boy, cat \rangle$.

This approximation is valid also for three-word phrases. When the matrices $\mathbf{S}$, $\mathbf{V}$, and $\mathbf{O}$ are such that $\mathbf{X}\mathbf{X}^T \approx \mathbf{I}$ with $\mathbf{X}$ one of the three matrices and $\mathbf{Y}\mathbf{X}^T \approx \mathbf{0}$ with $\mathbf{X}$ and $\mathbf{Y}$ two different matrices, Full Additive approximates a semantic convolution kernel comparing two sentences by summing the dot products of the words in the same role, that is,

$$\begin{aligned}
&\langle \mathbf{S}\,\vec{goats} + \mathbf{V}\,\vec{eat} + \mathbf{O}\,\vec{grass}, \mathbf{S}\,\vec{cows} + \mathbf{V}\,\vec{drink} + \mathbf{O}\vec{water} \rangle \approx \\
&\approx \langle goats, cows \rangle + \langle eat, drink \rangle + \langle grass, water \rangle
\end{aligned} \tag{14}$$

Results can again be easily extended to longer and different-length phrases.

Lexical Function Model. We distinguish composition with one- vs. two argument predicates. We illustrate the first through adjective–noun composition, where the adjective acts as the predicate, and the second with transitive verb constructions. Although we use the relevant syntactic labels, the formulas generalize to any construction with the same argument count. For adjective–noun phrases, the input is a sequence of (label, word) pairs ($x = ((A, w_1), (N, w_2))$) and the relevant decomposition set again includes only the single trivial decomposition into all the subparts: $\mathcal{R}(x) = \{((A, w_1), (N, w_2))\}$. The method itself is recursively defined as

$$f(x) = \begin{cases} f((A, w_1))f((N, w_2)) & \text{if } x \notin T_X = ((A, w_1), (N, w_2)) \\ \mathbf{W}_1 & \text{if } x \in T_x = (A, w_1) \\ \vec{w_2} & \text{if } x \in T_x = (N, w_2) \end{cases} \tag{15}$$

Here, K and g are, respectively, the dot and Frobenius product, $K_1(f(x),f(y)) = f(x)^T f(y)$, and $K_2(f(x),f(y)) = f(x)f(y)^T$. Using Equation (3), we have then

$$\begin{aligned} &K(f(\text{tall boy})), f(\text{red cat})) = \langle \mathbf{TALL}\, \vec{boy}, \mathbf{RED}\, \vec{cat} \rangle = \\ &= \langle \mathbf{TALL}^T \mathbf{RED}, \vec{boy}\, \vec{cat}^T \rangle_F = g(K_1(f(tall), f(red)), K_2(f(boy), f(cat))) \end{aligned} \quad (16)$$

The role of predicate and argument words in the final dot product is clearly separated, showing again the structure-sensitive nature of the decomposition of the comparison operations. In the two-place predicate case, again, the input is a set of (label, word) tuples, and the relevant decomposition set only includes the single trivial decomposition into all subparts. The CDSM f is defined as

$$f(x) = \begin{cases} f((S\, w_1)) \otimes f((V\, w_2)) \otimes f((O\, w_3)) & \text{if } x \notin T_X = ((S\, w_1)\,(V\, w_2)\,(O\, w_3)) \\ \vec{w} & \text{if } x \in T_X = (l\, w) \text{ and } l \text{ is } S \text{ or } O \\ \mathtt{W} & \text{if } x \in T_X = (V\, w) \end{cases} \quad (17)$$

K is the dot product and $g(x,y,z) = \langle x, y \otimes z \rangle_F$, $K_1(f(x),f(y)) = \sum_j (f(x) \otimes f(y))_j$—that is, the tensor contraction[2] along the second index of the tensor product between $f(x)$ and $f(y)$—and $K_2(f(x),f(y)) = K_3(f(x),f(y)) = f(x) \otimes f(y)$ are tensor products. The dot product of $\vec{goats}\, \mathtt{EAT}\, \vec{grass}$ and $\vec{cows}\, \mathtt{DRINK}\, \vec{water}$ is (by using Equation (4))

$$\begin{aligned} &K(f(((S\, goats)\,(V\, eat)\,(O\, grass))), f(((S\, cows)\,(V\, drink)\,(O\, water)))) = \\ &= \langle \vec{goats}\, \mathtt{EAT}\, \vec{grass}, \vec{cows}\, \mathtt{DRINK}\, \vec{water} \rangle = \\ &= \langle \textstyle\sum_j (\mathtt{EAT} \otimes \mathtt{DRINK})_j, \vec{goats} \otimes \vec{grass} \otimes \vec{cows} \otimes \vec{water} \rangle_F = \\ &= g(K_1(f((V\, eat)), f((V\, drink))), \\ &\qquad K_2(f((S\, goats)), f((S\, cows))) \otimes K_3(f((O\, grass)), f((O\, water)))) \end{aligned} \quad (18)$$

We rewrote the equation as a Frobenius product between two fourth-order tensors. The first combines the two third-order tensors of the verbs $\sum_j (\mathtt{EAT} \otimes \mathtt{DRINK})_j$ and the second combines the vectors representing the arguments of the verb, that is: $\vec{goats} \otimes \vec{grass} \otimes \vec{cows} \otimes \vec{water}$. In this case as well we can separate the role of predicate and argument types in the comparison computation.

Extension of the Lexical Function to structured objects of different lengths is treated by using the identity element ϵ for missing parts. As an example, we show here the comparison between *tall boy* and *cat* where the identity element is the identity matrix $\mathbf{I}$:

$$\begin{aligned} &K(f(\text{tall boy})), f(\text{cat})) = \langle \mathbf{TALL}\, \vec{boy}, \vec{cat} \rangle = \langle \mathbf{TALL}\, \vec{boy}, \mathbf{I}\, \vec{cat} \rangle = \\ &= \langle \mathbf{TALL}^T \mathbf{I}, \vec{boy}\, \vec{cat}^T \rangle_F = g(K_1(f(tall), f(\epsilon)), K_2(f(boy), f(cat))) \end{aligned} \quad (19)$$

Dual Space Model. We have until now applied the CC to linear CDSMs with the dot product as the final comparison operator (what we called K). The CC also holds for the effective Dual Space model of Turney (2012), which assumes that each word has two distributional representations, w_d in "domain" space and w_f in "function" space. The similarity of two phrases is directly computed as the geometric average of the separate similarities between the first and second words in both spaces. Even though

2 Grefenstette et al. (2013) first framed the Lexical Function in terms of tensor contraction.

there is no explicit composition step, it is still possible to put the model in CC form. Take $x = (x_1, x_2)$ and its trivial decomposition. Define, for a word w with vector representations w_d and w_f: $f(w) = \vec{w_d}\vec{w_f}^T$. Define also $K_1(f(x_1), f(y_1)) = \sqrt{\langle f(x_1), f(y_1)\rangle_F}$, $K_2(f(x_2), f(y_2)) = \sqrt{\langle f(x_2), f(y_2)\rangle_F}$ and $g(a, b)$ to be $\sqrt{ab}$. Then

$$\begin{aligned} & g(K_1(f(x_1), f(y_1)), K_2(f(x_2), f(y_2))) = \\ & = \sqrt{\sqrt{\langle \vec{x_{d1}}\vec{x_{f1}}^T, \vec{y_{d1}}\vec{y_{f1}}^T\rangle_F} \cdot \sqrt{\langle \vec{x_{d2}}\vec{x_{f2}}^T, \vec{y_{d2}}\vec{y_{f2}}^T\rangle_F}} = \\ & = \sqrt[4]{\langle \vec{x_{d1}}, \vec{y_{d1}}\rangle \cdot \langle \vec{x_{f1}}, \vec{y_{f1}}\rangle \cdot \langle \vec{x_{d2}}, \vec{y_{d2}}\rangle \cdot \langle \vec{x_{f2}}, \vec{y_{f2}}\rangle} = \\ & = \mathrm{geo}(\mathrm{sim}(x_{d1}, y_{d1}), \mathrm{sim}(x_{d2}, y_{d2}), \mathrm{sim}(x_{f1}, y_{f1}), \mathrm{sim}(x_{f2}, y_{f2})) \end{aligned} \tag{20}$$

A Neural-network-like Model. Consider the phrase $(w_1, w_2, \ldots, w_n)$ and the model defined by $f(x) = \sigma(\vec{w_1} + \vec{w_2} + \ldots + \vec{w_n})$, where $\sigma(\cdot)$ is a component-wise logistic function. Here we have a single trivial decomposition that includes all the subparts, and $\gamma(x_1, \ldots, x_n)$ is defined as $\sigma(x_1 + \ldots + x_n)$. To see that for this model the CC cannot hold, consider two two-word phrases $(a\, b)$ and $(c\, d)$

$$\begin{aligned} K(f((a,b)), f((c,d))) = \langle f((a,b)), f((c,d))\rangle = \textstyle\sum_i \left[\sigma(\vec{a} + \vec{b})\right]_i \cdot \left[\sigma(\vec{c} + \vec{d})\right]_i \\ = \quad \textstyle\sum_i \left(1 + e^{-a_i - b_i} + e^{-c_i - d_i} + e^{-a_i - b_i - c_i - d_i}\right)^{-1} \end{aligned} \tag{21}$$

We need to rewrite this as

$$g(K_1(\vec{a}, \vec{c}), K_2(\vec{b}, \vec{d})) \tag{22}$$

But there is no possible choice of g, K_1, and K_2 that allows Equation (21) to be written as Equation (22). This example can be regarded as a simplified version of the neural-network model of Socher et al. (2011). The fact that the CC does not apply to it suggests that it will not apply to other models in this family.

5. Conclusion

The Convolution Conjecture offers a general way to rewrite the phrase similarity computations of CDSMs by highlighting the role played by the subparts of a composed representation. This perspective allows for a better understanding of the exact operations that a composition model applies to its input. The Convolution Conjecture also suggests a strong connection between CDSMs and semantic convolution kernels. This link suggests that insights from the CDSM literature could be directly integrated in the development of convolution kernels, with all the benefits offered by this well-understood general machine-learning framework.

Acknowledgments

We thank the reviewers for helpful comments. Marco Baroni acknowledges ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Clark, Stephen. 2015. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd ed.* Blackwell, Malden, MA. In press.

Coecke, Bob, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Ganesalingam, Mohan and Aurélie Herbelot. 2013. Composing distributions: Mathematical structures and their linguistic interpretation. Working paper, Computer Laboratory, University of Cambridge. Available at `www.cl.cam.ac.uk/∼ah433/`.

Grefenstette, Edward, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. *Proceedings of IWCS*, pages 131–142, Potsdam.

Guevara, Emiliano. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala.

Haussler, David. 1999. Convolution kernels on discrete structures. Technical report USCS-CL-99-10, University of California at Santa Cruz.

Mehdad, Yashar, Alessandro Moschitti, and Fabio Massimo Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. In *Proceedings of NAACL*, pages 1,020–1,028, Los Angeles, CA.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

Socher, Richard, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada.

Turney, Peter. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Turney, Peter and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Zanzotto, Fabio Massimo, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1,263–1,271, Beijing.