Representation Learning for Type-Driven Composition

Gijs Wijnholds
Utrecht Institute of Linguistics OTS
Utrecht University
g.j.wijnholds@uu.nl

Mehrnoosh Sadrzadeh
Department of Computer Science
University College London
m.sadrzadeh@ucl.ac.uk

Stephen Clark
School of Electronic Engineering and Computer Science
Queen Mary University of London
stephen.clark609@gmail.com

Abstract

This paper is about learning word representations using grammatical type information. We use the syntactic types of Combinatory Categorial Grammar to develop multilinear representations, i.e. maps with n arguments, for words with different functional types. The multilinear maps of words compose with each other to form sentence representations. We extend the skipgram algorithm from vectors to multilinear maps to learn these representations and instantiate it on unary and binary maps for transitive verbs. These are evaluated on verb and sentence similarity and disambiguation tasks and a subset of the SICK relatedness dataset. Our model performs better than previous type-driven models and is competitive with state of the art representation learning methods such as BERT and neural sentence encoders.

1 Introduction

We develop a novel technique for learning word representations by using syntactic type information of the words to learn representations for them and the constituency-based structure of the sentence to compose the representations. The word representations are multilinear maps, i.e. maps with variable number of arguments, where the number of arguments and the type of each map come from the syntactic type of each word. The word representation are composed via the application and further composition of the results of these maps, based on constituency structure.

For instance, a noun such as *children* or *ball* is represented by a vector, i.e. $\overrightarrow{children}$ or $\overrightarrow{ball}$, which can be thought of as 0-linear maps as they have no input or output. An adjective such as *young* is represented by a unilinear map $\underline{young}$, i.e. a linear map of one argument, which at input takes an argument of type noun, e.g. *children* and at output returns an argument of type noun, i.e. *young children*. A transitive verb such as *play*, is represented by a bilinear map $\underline{play}$, i.e. a linear map with two arguments, which at input takes two arguments of type noun, e.g. *children* and *ball*, and at output returns an argument of type sentence, i.e. *young children play ball*. An adjective-noun phrase representation is obtained by applying the representation of the adjective to the representation of its noun, i.e. by applying the unilinear map of the adjective to the vector of the noun. A sentence representation is obtained by the composition of two applications, i.e. by first applying the representation of the verb, e.g. $\underline{play}$ to the representation of the object, e.g. $\overrightarrow{ball}$, resulting in a unilinear map for the representation of the verb phrase *play ball*, and subsequently applying this verb phrase to the representation of the subject, e.g. $\overrightarrow{children}$.

The representations are learnt by generalising the skipgram model with negative sampling of Mikolov et al. (2013) from vectors to higher order tensors, which are the intended multilinear maps. The types and composition operations come from the syntactic types and combinators of Combinatory Categorial Grammar (CCG) (Steedman, 2000). CCG is a phrase structure grammar formalism based on the combinatory logic of Curry and Feys (Curry and Feys, 1958). It assigns types defined in the notation of the combinatory logic to words of language and uses the operations of the combinatory logic to compose these types to obtain types for the phrases and sentences containing them. For instance, a word can have a CCG functional type of n arguments; this word will be represented in our setting by an n-ary map that uses the representations of its arguments, in a skipgram-style model, to predict the representations of the contexts of their composed phrases. As an example, consider a transitive verb; it has a CCG functional type of two arguments. Its representation is thus a binary map that predicts the contexts of its subject-verb-object phrases. Since the specific subject-verb-object phrases obtained may be sparse, we approximate the higher order maps with a set of lower order ones. As a result, a word with a CCG type of n arguments, gets represented by n maps of $n-1$ arguments; these transform the representations of a certain number of the arguments to predict the contexts of the remaining arguments. A transitive verb is now represented by two unary maps of one argument each; one of them transforms the object representation to predict its subject contexts, and the other transforms its subject representations to predict its object contexts. These lower order approximations are combined with each other to produce one single representation for the word with functional type.

Our generalised skipgram algorithm is modular, i.e. the skipgram model of Mikolov et al. (2013) and its extension to adjective matrices (Maillard and Clark, 2015) are special cases of it. We instantiate our model on binary and unary maps for transitive verbs. After learning these representations, we evaluate them on verb similarity, compositional sentence similarity and disambiguation tasks, and a subset of the SICK relatedness dataset (Marelli et al., 2014).

In the verb and sentence similarity and verb disambiguation datasets, our model outperforms all previous type-driven models, and in most cases it also outperforms InferSent and Universal Sentence encoders, as well as pre-trained ELMo and BERT embeddings. However, it does not outperform BERT embeddings fine-tuned on NLI data. In the subset of SICK, our model only outperforms all previous type-driven models. Despite that, our model is motivated by linguistic theory, is simple and quick to train, and has the potential for improvement (which we expand on in the conclusion).

Code and data to train representations and reproduce our work is available online.[1]

Background There is a plethora of methods for word embeddings, with few of them distinguishing the grammatical types of the words. For adjectives, we have the regression model of Baroni and Zamparelli (2010) that approximates the holistic adjective-noun vectors to learn adjective matrices; we also have the skipgram model of Maillard and Clark (2015) that learns a transformation between fixed vectors for nouns and adjective-noun combinations. The model of Grefenstette and Sadrzadeh (2011) takes the sum of the outer products of the vectors of subjects and objects, and the Kronecker product of the verb vector with itself, to learn verb matrices. Later work uses multi-step regression to learn a verb cube, i.e. a multidimensional array of depth 1, by iteratively approximating a holistic subject-verb and verb-object vector (Grefenstette et al., 2013). The model of Paperno et al. (2014) overcomes the sparsity issues of this technique and approximates the cubes by two matrices. The plausibility model of Polajnar et al. (2014), learns a verb matrix/cube by optimising a model that distinguishes between observed subject-verb-object triples and randomly generated ones. Our work is different from these, since we use a skipgram-style model rather than combining the subject and object vectors or the verb vectors, as done by Grefenstette and Sadrzadeh (2011), or performing regression, as done by Paperno et al. (2014) and Polajnar et al. (2014).

Sentence embeddings are either learnt by mixing word embeddings e.g. the additive models of (Mitchell and Lapata, 2010; Mikolov et al., 2013), or as a whole, e.g. the supervised InferSent (Conneau et al., 2017) and Universal Sentence Encoder (Cer et al., 2018), and the unsupervised ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) models. None, however, explicitly take into account grammatical information. Tree-RNNs (Socher et al., 2013), Tree-LSTMs (Tai et al., 2015), and Lifted Matrix Space model (Chung et al., 2018), do use the constituency tree of a sentence as a guide, but to learn a semantic function composition rather than different types of representations for words. Our work is different from these, since we start our learning procedure by taking the grammatical types of words into account and then compose these initially learnt representations with each other based on the structure of phrases they are part of, rather then by adding or learning different composition operators, or learning the entire phrase/sentence at once.

On the other hand, formal distributional models, e.g. the categorial framework of Coecke et al. (2010, 2013), the linear regression approach of Baroni et al. (2014), and the Combinatory Categorial Grammar (CCG) tensor contraction model of Maillard et al. (2014), directly take the grammatical types of words into account, but fail to scale up to sentences of any length and complexity, and do not

[1]`github.com/gijswijnholds/tensorskipgram-torch`

perform as well as their neural embedding counterparts. To remedy these issues, our model makes use of a simple neural network to learn the type-driven word representations in such a way that their composition leads to improved results.

2 Multilinear Skipgram Embeddings

The skipgram model with negative sampling generates word embeddings by optimising a logistic regression objective in which target vectors have high inner product with context vectors for positive contexts, and low inner product with negative ones. Given a target word n and a set of positive contexts C, a set of negative contexts $\overline{C}$ is sampled from a unigram distribution raised to some power (here: $3/4$, after Levy et al. (2015)).

Initially, both target vectors $\mathbf{n}$ and context vectors $\mathbf{c}$ are randomly intialised, and during training the model updates both target and context vectors to maximise the following objective function:

$$\sum_{c \in C} \log \sigma(\mathbf{n} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbf{n} \cdot \overline{\mathbf{c}}) \qquad (1)$$

We generalise the skipgram model following the typing of Combinatorial Categorial Grammar (CCG, Steedman (2000)). CCG has a transparent interface between syntax and semantics and robust wide-coverage parsers (Clark and Curran, 2007; Hockenmaier and Steedman, 2007). Syntactic types of CCG are either atomic, e.g. nouns/noun phrases: NP and sentences: S, or functional. Functional types are either of the form Y/X or $Y \backslash X$; they take an argument of type X and return an argument of type Y, where for $\backslash$ the argument occurs to the left and for $/$ it occurs to the right. Examples of functional types are adjectives: NP/NP, intransitive verbs: $S \backslash NP$ and transitive verbs: $(S \backslash NP)/NP$.

Types are composed with each other through the combinatorial rules of CCG, which include forward and backward application and composition, type-raising, and backward-cross and forward-cross composition. An example of forward application is when an adjective composes with a noun, producing a noun. An example of backward application is when a verb phrase composes with a noun phrase producing a sentence.

$$\frac{NP/NP \quad NP}{NP}> \qquad \frac{NP \quad S\backslash NP}{S}<$$

Forward and backward composition are used in composing auxiliary phrases; cross composition and the type-raising combinators are used in cases of coordination and gapping.

Following the tensor semantics of CCG, developed in Maillard et al. (2014), in our model, we represent a word W with a functional type of n arguments by a n-ary map $\mathbb{W}$ from the argument spaces to the result space:

$$\mathbb{W}_{(n)} \colon V_1 \times ... \times V_n \to V_{n+1}$$

where V_i's are (finite dimensional) vector spaces over the field of reals and the subscript n denotes the arity of the map $\mathbb{W}$. Equivalently, $\mathbb{W}_{(n)}$ is an $(n+1)$th-order tensor $\mathbb{W}_{i_1 \dots i_{n+1}}$ in the space $V_1 \otimes ... \otimes V_n \otimes V_{n+1}$. Given a functional word W of n arguments and representations $\mathbf{d}_1, ..., \mathbf{d}_n$ of its arguments, we denote by $\mathbb{W}_{(n)}\mathbf{d}_1 ... \mathbf{d}_n$ the application of the representation of W to its arguments' representations. The model that learns the maps has the following objective function:

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

When W is a noun, $\mathbb{W}_{(0)}$ is a 0-ary map, equivalent to $\mathbb{W}_{i_1}$: a 1st-order tensor, i.e., a vector. In this case, the objective function reduces to the original skipgram model of Equation 1. For W an adjective, $\mathbb{W}_{(1)}$ is a unary map, equivalent to $\mathbb{W}_{i_1 i_2}$: a 2-nd order tensor, i.e., a matrix. The objective function that learns it was developed in Maillard and Clark (2015), and is as follows:

$$\sum_{c \in C} \log \sigma(\mathbb{W}_{(1)}\mathbf{d}_1 \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbb{W}_{(1)}\mathbf{d}_1 \cdot \overline{\mathbf{c}})$$

Since adjective-noun combinations are themselves nouns, taking the original skipgram linear context window will produce sensible adjective representations. This is however not the case for all words with functional types: for verbs, for example, subjects and objects may not be directly adjacent to the verb in a sentence and so one needs to commit to a full sentential context, leading to uninformative training data. Aside to that, training of a cube leads to over parameterisation. To overcome these issues simultaneously, we define lower order approximations of higher order tensors, where one argument of the functional type is left out of the composition and used as context.

More formally, a word W with a functional type of n arguments, is approximated by n maps of

$n-1$ arguments. Equivalently, in tensor form, we are approximating a *full* $n+1$th-order tensor, by decomposing it into n separate *partial* tensors of one lower order each. We denote the map equivalent of these partial tensors by $\widetilde{\mathbb{W}}^{i}_{(n-1)}$. The objective function of the model thus become as follows:

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

Here, d_i is a word with representation $\mathbf{d}_i$: an *observed* argument of W, and D_i is the set of all such arguments. Whereas, $\overline{d}_i$ is a word with representation $\overline{\mathbf{d}}_i$, which can in principle serve as an argument of W, but it is randomly sampled, so it is an *unobserved* argument of W. Similarly, $\overline{D}_i$ is the set of all such arguments.

We can decrease the order of the tensors even more by parameterising over subsets of contexts. For a word W of n arguments, when including 1 to $i \leq n$ of its n arguments in the context, we obtain an $(n-i)$th-order tensor, with the equivalent map $\widetilde{\mathbb{W}}^{1...i|i+1...n}_{(n-i+1)}$. The application of this map to the remaining $i+1$ to n arguments is $\widetilde{\mathbb{W}}^{1...i|i+1...n}_{(n-i+1)} \mathbf{d}_{i+1} ... \mathbf{d}_n$. So we predict as context the $1...i$th arguments by composing with the vectors for the $i+1...n$th arguments. We write $\widetilde{\mathbb{W}}^{1...i}_{(n-i+1)}$ when $i = n$, i.e. we use all arguments as context.

2.1 Instantiation to Verb Skipgram

We instantiate our model on transitive verbs. A transitive verb V has CCG type $(S\backslash NP)/NP$. Our *full* model learns a binary map $\mathbb{V}_{(2)}: V_1 \times V_2 \to V_3$, equivalent to a 3rd-order tensor, i.e. a cube, $\mathbb{V}_{i_1 i_2 i_3}$ to represent V. We denote d_1, i.e. the object of the verb, by o, its vector by $\mathbf{o}$, and d_2, i.e. its subject, by s, its vector by $\mathbf{s}$. The objective function of our *full* model for V is thus

$$\sum_{c \in C} \log \sigma(\mathbb{V}_{(2)} \mathbf{os} \cdot \mathbf{c}) + \sum_{\overline{c} \in \overline{C}} \log \sigma(-\mathbb{V}_{(2)} \mathbf{os} \cdot \overline{\mathbf{c}}) \quad (2)$$

We approximate $\mathbb{V}_{(2)}$ by training two unary maps, an object one $\widetilde{\mathbb{V}}^{1|2}_{(1)}$, which we denote by $\widetilde{\mathbb{V}}^{o|s}_{(1)}$, and a subject one $\widetilde{\mathbb{V}}^{2|1}_{(1)}$, which we denote by $\widetilde{\mathbb{V}}^{s|o}_{(1)}$. The map $\widetilde{\mathbb{V}}^{o|s}_{(1)}$ predicts the object of the verb, given a fixed subject; it is learnt as follows:

Representation	Order	Context
$\mathbb{V}^{sent\|o,s}_{(2)}$	binary	sentence
$\widetilde{\mathbb{V}}^{o\|s}_{(1)}/\widetilde{\mathbb{V}}^{s\|o}_{(1)}$	unary	obj/sbj
$\widetilde{\mathbb{V}}^{o}_{(0)}/\widetilde{\mathbb{V}}^{s}_{(0)}/\widetilde{\mathbb{V}}^{o,s}_{(0)}$	0-ary	obj/sbj/both
$\mathbb{V}^{sent\|s}_{(1)}$, $\mathbb{V}^{sent\|o}_{(1)}$	unary	sentence
$\mathbf{v}_{\text{skip}}$	0-ary	linear window

Table 1: Verb representations, ranging from 0-ary (vectors) to binary maps (cubes).

$$\sum_{o \in O} \log \sigma(\widetilde{\mathbb{V}}^{o|s}_{(1)} \mathbf{s} \cdot \mathbf{o}) + \sum_{\overline{o} \in \overline{O}} \log \sigma(-\widetilde{\mathbb{V}}^{o|s}_{(1)} \mathbf{s} \cdot \overline{\mathbf{o}}) \quad (3)$$

The map $\widetilde{\mathbb{V}}^{s|o}_{(1)}$ predicts the subject of the verb, given a fixed object, and is learnt as follows:

$$\sum_{s \in S} \log \sigma(\widetilde{\mathbb{V}}^{s|o}_{(1)} \mathbf{o} \cdot \mathbf{s}) + \sum_{\overline{s} \in \overline{S}} \log \sigma(-\widetilde{\mathbb{V}}^{s|o}_{(1)} \mathbf{o} \cdot \overline{\mathbf{s}}) \quad (4)$$

Here, S and O are the sets of observed subjects and objects of V, and $\overline{S}$ and $\overline{O}$ are the sets of V's unobserved subjects and objects.

We push the approximation one level further to also produce three 0-ary maps, i.e. vectors, for the verb. We denote these by $\widetilde{\mathbb{V}}^{o}_{(0)}$, $\widetilde{\mathbb{V}}^{s}_{(0)}$, $\widetilde{\mathbb{V}}^{o,s}_{(0)}$; they respectively represent a verb vector by only considering its objects, subjects, or both as context. These vectors are similar to the dependency based embeddings of Levy and Goldberg (2014).

We summarise all trained models by the arity of their maps and the choice of their contexts in Table 1. As baselines, we additionally train unary maps $\mathbb{V}^{sent|s}_{(1)}$ and $\mathbb{V}^{sent|o}_{(1)}$, which predict a full sentence context given the subject or object of the verb, and $\mathbf{v}_{\text{skip}}$ for the original skipgram vector of the verb.

2.2 Fusion

We consider two ways of combining our unary map skipgram verb representations into a single representation: the *middle* and *late* fusion methods of Bruni et al. (2014). Middle fusion takes a weighted average of the two verb representations, using the result to compute similarity scores. Late fusion uses each representation to compute separate similarity scores and then averages the results. Given a weighted average $M_\alpha(A, B) = \alpha A + (1-\alpha)B$ for $\alpha \in [0..1]$, and V, W two verbs, with approximated

Metric	Formula
vecsim	$\cos(\mathbf{a}, \mathbf{b}) = \dfrac{\mathbf{a} \cdot \mathbf{b}}{\lvert\mathbf{a}\rvert \lvert\mathbf{b}\rvert}$
matsim^S	$\underset{\mathbf{s} \in \mathcal{S}}{\text{med}} \cos(\widetilde{\mathbb{V}}_{(1)}\mathbf{s}, \widetilde{\mathbb{W}}_{(1)}\mathbf{s})$
matsim^O	$\underset{\mathbf{o} \in \mathcal{O}}{\text{med}} \cos(\widetilde{\mathbb{V}}_{(1)}\mathbf{o}, \widetilde{\mathbb{W}}_{(1)}\mathbf{o})$
cubesim	$\underset{\langle \mathbf{s}, \mathbf{o} \rangle \in \mathcal{A}}{\text{med}} \cos(\mathbb{V}_{(2)}\mathbf{os}, \mathbb{W}_{(2)}\mathbf{os})$

Table 2: Similarity metrics on vectors, matrices and cubes, based on clustering centroids.

subject and object matrices $\widetilde{\mathbb{V}}, \widetilde{\mathbb{V}}'$ and $\widetilde{\mathbb{W}}, \widetilde{\mathbb{W}}'$ the middle and late fusion operations are:

$$\text{mid} \qquad \text{sim}(M_\alpha(\widetilde{\mathbb{V}}, \widetilde{\mathbb{V}}'), M_\alpha(\widetilde{\mathbb{W}}, \widetilde{\mathbb{W}}')) \tag{5}$$

$$\text{late} \qquad M_\alpha(\text{sim}(\widetilde{\mathbb{V}}, \widetilde{\mathbb{W}}), \text{sim}(\widetilde{\mathbb{V}}', \widetilde{\mathbb{W}}')) \tag{6}$$

The same fusion methods are used in the compositional tasks, where either verb matrices are averaged before composition, or cosine scores are averaged after.

2.3 Clustering

In their adjective skipgram model, Maillard and Clark (2015) argued that cosine similarity, while suitable for vectors, does not capture any information about the function of matrices as unary maps and that instead one should measure how similarly the maps transform their arguments. The same holds for generalisations of unary maps to n-ary ones, equivalently, for matrices to higher order tensors. Following Maillard and Clark, we apply clustering to achieve this. The degree of similarity between two words W and W', each with a functional type of n arguments, is obtained by taking the median of the degrees of similarities of the applications of their maps $\mathbb{W}_{(n)}$ and $\mathbb{W}'_{(n)}$ on the clusters of their arguments. Since going through all the instantiations of the arguments is expensive, we cluster the most frequent argument vectors and work with the similarity between the two transformations applied to the centroids of each cluster. The resulting similarity function is defined as follows, for $\mathcal{D}$ the set of tuples of cluster centroids:

$$\text{tensorsim}: \underset{\langle \mathbf{d}_1, \ldots, \mathbf{d}_n \rangle \in \mathcal{D}}{\text{med}} \cos(\mathbb{W}_{(n)}\mathbf{d}_1 \ldots \mathbf{d}_n, \mathbb{W}'_{(n)}\mathbf{d}_1 \ldots \mathbf{d}_n)$$

Model type	Formula
Middle	$T(\mathbf{s}, M_\alpha(\widetilde{\mathbb{V}}_{(1)}, \widetilde{\mathbb{V}}'_{(1)}), \mathbf{o})$
Late	$M_\alpha(T(\mathbf{s}, \widetilde{\mathbb{V}}_{(1)}, \mathbf{o}), T(\mathbf{s}, \widetilde{\mathbb{V}}'_{(1)}, \mathbf{o}))$
Two	$M_\alpha(T_s(\mathbf{s}, \widetilde{\mathbb{V}}_{(1)}, \mathbf{o}), T_o(\mathbf{s}, \widetilde{\mathbb{V}}'_{(1)}, \mathbf{o}))$
Cube	$\mathbb{V}_{(2)}\mathbf{os}$

Table 3: Building Representations for Transitive Sentences. T represents a n-ary map composition model for transitive sentences, T_s is subject-directed composition, T_o is object-directed composition. When $\alpha = 0$ or $\alpha = 1$, the models reduce to the case of using one of the two verb matrix embeddings.

For the case of transitive verbs, we are dealing with binary map transformations and the above definition simplifies to considering the most frequent subjects and objects of the verb, clustering them separately, then applying the map to the centroid vectors and taking the median. The details of the different map transformation similarities that we obtain for transitive verbs using our model are given in Table 2.

3 Implementation and Evaluation

3.1 Implementation

We implemented all models in Python, using the `tensorflow` package (Abadi et al., 2016)[2]. Vectors were 100-dimensional; unary and binary maps, i.e. matrices and cubes, were shaped accordingly. The functional type-driven information was extracted from a dependency parsed corpus[3] containing ca.130M sentences and ca. 3.2B words, on which the initial regular noun vectors were also trained.

In the case of matrices and cubes with full sentential contexts, a pair of networks was trained separately for each verb, sharing the context matrix from the noun skipgram model. For the matrices with subject (resp. object) contexts, we trained a pair of networks (a subject network and an object network), each with a single embedding layer encoding all the verbs. In these networks, the context matrix consists of all possible object (resp. subject) context vectors. Here we considered both a fixed context matrix (from the noun skipgram model) and a trainable context matrix and found that the

[2]Our code was later changed to Pytorch.

[3]UKWaCkypedia, wacky.sslmit.unibo.it

trainable context matrix gave the best results[4], so we work with the latter. Negative samples were drawn from the distribution over objects/subjects of all verbs in the case of the partial tensor models. We considered $k = 10$ negative samples per subject/object.

3.2 Evaluation and Datasets

We evaluate our verb representations on four types of tasks: verb similarity, verb disambiguation, sentence similarity, including SVO sentences and SVO sentences with elliptical phrases, and a subset of the SICK sentence relatedness task.

3.3 Verb Similarity

We considered five verb similarity datasets of varying size: pairs of words from the MEN (Bruni et al., 2012) and SimLex-999 (Hill et al., 2015) datasets that were labelled as verbs, obtaining 22 and 222 verb similarity pairs, respectively. Next to these partial datasets, we considered VerbSim (Yang and Powers, 2006), a dataset of 130 verb pairs, and the more recent SimVerb-3500 dataset (Gerz et al., 2016), containing 3500 verb pairs.

3.4 Verb Disambiguation and Sentence Similarity

We considered seven tasks. (1,2) The two datasets introduced by Mitchell and Lapata (2008, 2010), dubbed ML08 and ML10. These datasets contain pairs of intransitive sentences; the 2008 dataset aims to disambiguate the verb of each sentence, the 2010 dataset is for computing sentence similarity. (3,4) The transitive verb disambiguation datasets of Grefenstette and Sadrzadeh (2011) (GS11) and Kartsaklis and Sadrzadeh (2013) (KS13a), and (5) the transitive sentence similarity dataset of Kartsaklis et al. (2013) (KS13b). (6,7) We additionally test on two recent datasets (Wijnholds and Sadrzadeh, 2019) (ELLDIS and ELLSIM), which extend the KS13a and KS13b datasets to sentences with verb phrase ellipsis in them.

The datasets ML08 and ML10, respectively, contain pairs of subject-verb, and verb-object phrases. Next to the additive baseline, we apply the unary map representations of verbs to the subject (or object) to get sentence representations: $\widetilde{\mathbb{V}}_{(1)}\mathbf{s}$ for subject-verb phrases, $\widetilde{\mathbb{V}}_{(1)}\mathbf{o}$ for verb-object phrases. For the separate subject-verb and verb-object maps, we apply middle and late fusion. To model a transitive sentence of the form *subj verb obj*, we compare verb-only and additive baselines with n-ary map models as described in Table 3. In the Two model in this table, we first apply $\widetilde{\mathbb{V}}^o_{(1)}$ to the subject vector, then mix it with the application of $\widetilde{\mathbb{V}}^s_{(1)}$ to the object vector. We then mix in the object/subject vectors and obtain three different models: **CA** for Copy Argument, **CAS** for Copy Argument Sum and **CATA** for Categorical Argument; see Table 4.

Model	Formula
CA	$M_\alpha(\mathbf{s}^T\widetilde{\mathbb{V}}_{(1)} \odot \mathbf{o}, \widetilde{\mathbb{V}}'_{(1)}\mathbf{o} \odot \mathbf{s})$
CAS	$M_\alpha(\mathbf{s}^T\widetilde{\mathbb{V}}_{(1)} + \mathbf{o}, \widetilde{\mathbb{V}}'_{(1)}\mathbf{o} + \mathbf{s})$
CATA	$M_\alpha(\mathbf{s}^T\widetilde{\mathbb{V}}_{(1)}, \widetilde{\mathbb{V}}'_{(1)}\mathbf{o})$

Table 4: Two-map models. We compose partial sentence embeddings using the subject- and object-directed verb matrix, and merge the two embeddings into one. M_α is the mixing operator defined before.

The ELLDIS and ELLSIM datasets of Wijnholds and Sadrzadeh (2019) contain sentences of the form *subj verb obj and subj* does too*. We first resolve the ellipsis by replacing the marker *does too* with its antecedent *verb object*, then apply a transitive model to the resulting *subj verb obj* and *subj* verb object* conjunct and finally combine the representations by addition; formally

$$E(\mathbf{s}, \widetilde{\mathbb{V}}_{(1)}, \mathbf{o}, \mathbf{s}^*) = T(\mathbf{s}, \widetilde{\mathbb{V}}_{(1)}, \mathbf{o}) + T(\mathbf{s}^*, \widetilde{\mathbb{V}}_{(1)}, \mathbf{o})$$

where $\mathbf{s}^*$ is the representation of *subj**.

3.5 SICK-R

The SICK relatedness task of Marelli et al. (2014) contains sentence pairs that are scored between 1 and 5 on semantic relatedness to evaluate compositional distributional models for relatedness. To evaluate our verb representations, we extract the verbs with their arguments (subjects and/or objects) from dependency parsed sentences, use one of the previously described composition models to generate a single verb representation for the verb-argument tuple, and compose this with the vectors for the remaining words in the

[4]We argue that this is because contexts in the noun skip-gram model are more general as they serve as contexts to many different target words.

	MEN$_v$	**SL**$_v$	**VS**	**SV**$_d$	**SV**$_t$
$\mathbf{v}_{\textbf{skip}}$	0.28	0.05	0.34	0.22	0.18
$\mathbf{V}_{Kron}$	0.38	0.1	0.37	0.22	0.18
$\mathbf{V}_{Rel}$	0.33	0.05	0.34	0.22	0.18
$\widetilde{\mathbb{V}}^{o/s/s,o}_{(0)}$	0.25	0.27	**0.56**	0.25	0.20
$\mathbb{V}^{sent\|s/o}_{(1)}$	0.50	0.16	0.09	-0.02	0.02
$\widetilde{\mathbb{V}}^{o\|s/s\|o}_{(1)}$	**0.59**	**0.34**	0.55	**0.29**	**0.24**
$\mathbb{V}^{sent\|s,o}_{(2)}$	0.04	0.02	-0.08	-0.01	-0.03
SoTA	n/a	0.39	0.65	0.40	0.30

Table 5: Spearman ρ correlation on verb similarity datasets. The subscript v indicates that we are looking at the partial verb-only dataset. For SimVerb we distinguish between the development and test set. State of the art scores are taken from (Chersoni et al. (2016), VS) and (Gerz et al. (2016), SL$_v$, SV$_d$, SV$_t$). For MEN, we did not find any results on the verb subset.

	ML08	**ML10**	**GS11**	**KS13a**	**KS13b**
$\mathbf{v}_{\textbf{skip}}$	0.07	0.40	0.23	0.18	0.45
$\mathbf{V}_{Kron}$	**0.25**	0.40	0.27	**0.26**	0.45
$\mathbf{V}_{Rel}$	0.11	0.43	0.31	0.18	0.47
$\widetilde{\mathbb{V}}^{o/s/s,o}_{(0)}$	0.06	0.53	0.33	0.10	0.64
$\mathbb{V}^{sent\|s/o}_{(1)}$	0.16	-0.00	0.37	0.06	-0.06
$\widetilde{\mathbb{V}}^{o\|s/s\|o}_{(1)}$	0.12	**0.64**	**0.40**	0.22	**0.69**
$\mathbb{V}^{sent\|s,o}_{(2)}$	0.18	0.00	-0.03	0.00	-0.03
SoTA	0.19	0.45	0.46	0.22	0.73
Human	0.66	0.71	0.74	0.58	0.75

Table 6: Spearman ρ correlation of verbs of SVO sentence level tasks. Each score is a maximum score out of possible clusters and fusion weights. State of the art scores are taken from (Mitchell and Lapata (2008),ML08), (Milajevs et al. (2014),GS11,KS13b) and (Kartsaklis and Sadrzadeh (2013),ML10,KS13a).

sentence. We used the Spacy[5] parser combined with a postprocessing script to correct cases of coordination of verbs and arguments, as we expected this to be vital information in the dataset. To keep this process manageable, we used the SemEval subset of the SICK dataset. We evaluate our best performing verb unary map representations ($\widetilde{\mathbb{V}}^{o|s/s|o}_{(1)}$), as well as the two analytical verb representations $\mathbf{V}_{Kron}$ and $\mathbf{V}_{Rel}$.

3.6 Comparison with Other Methods

At the verb level, we compare our skipgram verb representations (Table 1) with two verb representation methods from the type-driven literature (Grefenstette and Sadrzadeh, 2011). The first representation, referred to by *Kronecker*, lifts a verb vector to a matrix representation using outer product. The second representation is the *Relational* model, where a verb matrix is taken to be the sum of the outer products of its subject and object vectors; formally:

$$\mathbf{V}_{Kron} = \mathbf{v}_a \otimes \mathbf{v}_a \qquad \mathbf{V}_{Rel} = \sum_i \mathbf{s}_i \otimes \mathbf{o}_i$$

At the sentence level, we compare our model with that of Mitchell and Lapata (2010), which given a sentence adds the vectors of the words therein, and also with supervised sentence encoders, InferSent (Conneau et al., 2017), as well as, Universal Sentence Encoder (Cer et al., 2018). For these latter, we take off-the-shelf encoders to map the sentence pairs in our evaluation datasets to a pair of embeddings, and compute the cosine similarity between these. We moreover compare to state-of-the-art contextualised encoders ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). For ELMo, we use a pre-trained model and apply mean pooling[6]. For BERT, we take the implementation of Reimers and Gurevych (2019)[7], as it implements both the original pre-trained BERT models and fine-tuned sentence embedding models. To this, we apply max, mean, and CLS token pooling, and report the best scores out of all models and pooling types, for the pre-trained models and the fine-tuned models.

4 Results

4.1 Verb Level Tasks

The correlation results on verb similarity tasks are displayed in Table 5. Here, for the case of verb vectors, the general skipgram model is outperformed by the vectors trained using our partial model on the verb arguments as context, and in fact these show the highest performance on the VerbSim dataset. That the unary and binary maps representations with the full sentence as context perform rather

[5]https://spacy.io/

[6]https://tfhub.dev/google/elmo/2

[7]https://github.com/UKPLab/sentence-transformers

	ML08	ML10	GS11	KS13a	KS13b
$C(+)$	0.17	0.54	0.19	0.18	0.67
$C(\mathbf{V}_{Kron})$	0.08	0.40	0.20	0.28	0.53
$C(\mathbf{V}_{Rel})$	0.19	0.51	0.32	0.19	0.51
$C(\mathbb{V}^{sent\|s/o}_{(1)})$	-0.04	0.00	0.25	0.20	0.54
$C(\widetilde{\mathbb{V}}^{o\|s/s\|o}_{(1)})$	**0.19**	**0.55**	**0.54**	**0.37**	**0.75**
$C(\mathbb{V}^{sent\|s,o}_{(2)})$	—	—	-0.02	-0.04	0.06
Human	0.66	0.71	0.74	0.58	0.75

Table 7: Spearman ρ scores on compositional tasks. $C(+)$ denotes the additive model, whereas the other rows represent the best score for compositional models with different verb representations.

	ML08	ML10	GS11	KS13a	KS13b
$\widetilde{\mathbb{V}}^{o\|s/s\|o}_{(1)}$	0.19	0.55	0.54	**0.37**	0.75
IS	0.18	0.63	0.30	0.17	0.78
USE	0.04	0.33	0.09	0.21	0.54
ELMo	0.17	0.54	0.11	0.24	0.73
BERT$_p$	0.19	0.34	0.24	0.32	0.61
BERT$_f$	**0.32**	**0.74**	**0.61**	0.32	**0.82**
Human	0.66	0.71	0.74	0.58	0.75

Table 8: Spearman ρ scores on compositional tasks, for our proposed unary map verb representation versus state of the art sentence embedding methods.

poorly, and in many cases worse than the vector representations, illustrates that the choice of context is too general for these higher-order representations. On four out of the five tasks, our approximated models that train unary maps with a restricted notion of context, outperform all other models: the most significant of these increases are for the 3000 entry test subset of the SimVerb dataset: here we observe an increase from 0.18 to 0.24.

Table 6 shows the correlation scores on the verbs of the SVO sentence level tasks. In this experiment, we perform the sentence disambiguation and similarity tasks by only using the verbs of the sentences. We observe the same pattern in the results: training verb vectors on dependency label contexts slightly improves the performance. This is against the erratic performance of the binary map representations (on all but the ML2008 dataset). Again, our approximated unary map representations with a restricted context significantly outperforms the other methods.

In the majority of the verb similarity datasets we do not improve the state of the art, but in the majority of the verb parts of the SVO sentence datasets, we do.

4.2 Sentence Level Tasks

4.2.1 Verb Disambiguation and SVO Sentence Similarity Datasets

The most interesting results, however, come from the SVO sentence tasks. These compute a representation for each sentence of the dataset by composing the representations of the words of that sentence, rather than by only working with individual word representations, as was done in the previous two tasks. Table 7 contrasts the additive models (top row), type-driven methods that use the Kronecker (second row) and Relational (third row) verb representations, against the type-driven model that uses skipgram representations (resp. full context binary maps, full context unary maps, restricted context unary maps).

While the skipgram binary map verb representations with full sentences as context perform slightly better in a sentence context, they generally underperform the additive baseline and the non-skipgram tensors. We argue that this is mainly due to the choice of context: the full sentence doesn't tell us enough about the subjects and objects of the verb, whereas the Relational model directly encodes this information. Similarly to the verb similarity results, the binary map representations show a very poor performance, which we argue is due to data sparsity. Even though the binary map implicitly model properties of arguments of the verbs, their representation is too sparse to effectively model anything. Our proposed unary map model remedies both the sparsity problem and the choice of context, and outperforms all the other representations, save on the ML2008 dataset. This model also improves the state of the art in all the datasets.

4.2.2 Elliptical Phrase and SICK Datasets

The results in Tables 9 show that our proposed verb unary map representations achieve competitive results compared to the additive baseline, and pre-trained BERT embeddings, on the ELLDIS and ELLSIM tasks and on (a subset of) the SICK relatedness task. What is more, they clearly outperform the analytic tensors and in ellipsis datasets; they also improve the state of the art of ELLDIS, which

Add	**Kron**	**Rel**	$\widetilde{\mathbb{V}}^{o\|s/s\|o}_{(1)}$	**IS**	**USE**	**BERT**$_p$	**BERT**$_f$
0.31	0.30	0.37	0.56	0.34	0.27	0.36	**0.65**
0.67	0.52	0.65	0.76	**0.80**	0.68	0.67	0.79
0.71	0.58	0.44	0.70	0.74	**0.76**	0.70	**0.76**

Table 9: Spearman ρ scores on the ELLDIS (top), ELLSIM (middle), and SICK relatedness (bottom) tasks.

was 0.53, and provide equal results to the state of the art of ELLSIM, which was 0.76 , both reported in Wijnholds and Sadrzadeh (2019). However, they are surpassed by fine-tuned BERT sentence embeddings and sentence encoders, that achieve the highest. For SICK, to verify that the high performance of our verb maps is not caused simply by adding in the vectors for the remaining word of a sentence, we did an ablation in which the rest of the sentence was not considered. Using addition of vectors, this gave a ρ of 0.61, and for the compositional verb matrices this gave 0.62 (cf. 0.71 and 0.70 in Table 9).

4.3 Comparison with Sentence Embeddings

We compare our model with the InferSent encoder and the Universal Sentence Encoder, and with ELMo and BERT encodings in Table 8. Although our embeddings outperform Universal Sentence Encoder on all tasks, on the ML2010 and KS2014 dataset InferSent performs higher, possibly due to its high embedding dimensionality (4096). For the BERT embeddings we observe an interesting pattern: our proposed method outperforms any pre-trained BERT model, but after fine-tuning on NLI datasets, the BERT models score the highest on all datasets but KS2013. Although fully analysing the syntactic awareness of BERT is beyond the scope of this paper, it seems that both explicitly modelling syntax in the embeddings as our method does, and fine-tuning BERT embeddings are viable strategies.

5 Conclusion

We generalised the skipgram model (Mikolov et al., 2013) to learn multilinear map representations for words with functional types using the setting of Combinatory Categorial Grammar. Our model reduces to the original skipgram for atomic types such as nouns, and to the adjective skipgram model of Maillard and Clark (2015), for functional types of one argument. To overcome potential sparsity issues we approximated higher arity maps with a set of lower arity ones and showed that such approximations provide better results.

The model was implemented on transitive verbs, learning binary and a set of approximated unary representations. These were evaluated on verb similarity and disambiguation and sentence similarity tasks. The unary map approximations significantly outperformed previous type-driven verb representations. They also outperformed sentence encoders and pre-trained BERT embeddings. When moving to datasets of longer sentences, e.g. sentences with elliptical phrases and the SICK relatedness, some sentence encoders and fine-tuned BERT representations were superior.

Our multilinear skipgram model paves the way for a new generation of type-driven representations, in line with recent research highlighting benefits of syntactic biases injected into representation learning (Kuncoro et al., 2020). Furthermore, our model is fast to train, guided by a linguistic calculus (CCG), and produces syntax-aware sentence embeddings. Performance could potentially be improved by adding non-linearities to the model, as in Socher et al. (2013) and by modelling complex syntactic phenomena such as auxiliaries and negation.

Acknowledgments

The authors gratefully acknowledge a multitude of reviewers for their careful efforts in reviewing this work. Wijnholds is grateful for receiving PGR student funding from the School of Electronic Engineering and Computer Science at Queen Mary University of London, and is currently supported by the Dutch Research Council (NWO) under the scope of the project "A composition calculus for vector-based semantic modelling with a localization for Dutch" (360-89-070). Sadrzadeh acknowledges the Royal Academy of Engineering Industrial Fellowship IF192058.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. *LiLT (Linguistic Issues in Language Technology)*, 9.

Marco Baroni and Roberto Zamparelli. 2010. 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. Association for Computational Linguistics.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2016. Representing verbs with rich contexts: an evaluation on verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1967–1972, Austin, Texas. Association for Computational Linguistics.

WooJin Chung, Sheng-Fu Wang, and Samuel Bowman. 2018. The lifted matrix-space model for semantic composition. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 508–518, Brussels, Belgium. Association for Computational Linguistics.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Bob Coecke, Edward Grefenstette, and Mehrnoosh Sadrzadeh. 2013. Lambek vs. Lambek: Functorial vector space semantics and string diagrams for Lambek calculus. *Annals of Pure and Applied Logic*, 164(11):1079 – 1100. Special issue on Seventh Workshop on Games for Logic and Programming Languages (GaLoP VII).

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36(1):345–384.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Haskell B. Curry and Richard Feys. 1958. *Combinatory Logic*. North-Holland, Amsterdam.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. 2016. SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas. Association for Computational Linguistics.

E. Grefenstette, G. Dinu, Y. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 131–142, Potsdam, Germany. Association for Computational Linguistics.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2013. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601, Seattle, Washington, USA. Association for Computational Linguistics.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013. Separating disambiguation from composition in distributional semantics. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 114–123, Sofia, Bulgaria. Association for Computational Linguistics.

Adhiguna Kuncoro, Lingpeng Kong, Daniel Fried, Dani Yogatama, Laura Rimell, Chris Dyer, and Phil Blunsom. 2020. Syntactic structure distillation pretraining for bidirectional encoders. *arXiv preprint arXiv:2005.13482*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland. Association for Computational Linguistics.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

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

Jean Maillard, Stephen Clark, and Edward Grefenstette. 2014. A type-driven tensor-based semantics for CCG. In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pages 46–54, Gothenburg, Sweden. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 216–223, Reykjavik, Iceland. European Languages Resources Association (ELRA).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 708–719.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 90–99, Baltimore, Maryland. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2014. Using sentence plausibility to learn the semantics of transitive verbs. *Learning Semantics Workshop, NIPS*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mark Steedman. 2000. *The syntactic process*, volume 24. MIT Press.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.

Gijs Wijnholds and Mehrnoosh Sadrzadeh. 2019. Evaluating composition models for verb phrase elliptical sentence embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 261–271, Minneapolis, Minnesota. Association for Computational Linguistics.

Dongqiang Yang and David Martin Powers. 2006. Verb similarity on the taxonomy of wordnet. In *Pro-*

ceedings of the Third International WordNet Conference GWC 2006, South Jeju Island, Korea, pages 121–128. Masaryk University.