Learning Semantically and Additively Compositional Distributional Representations

Ran Tian and **Naoaki Okazaki** and **Kentaro Inui**
Tohoku University, Japan
{tianran, okazaki, inui}@ecei.tohoku.ac.jp

Abstract

This paper connects a vector-based composition model to a formal semantics, the Dependency-based Compositional Semantics (DCS). We show theoretical evidence that the vector compositions in our model conform to the logic of DCS. Experimentally, we show that vector-based composition brings a strong ability to calculate similar phrases as similar vectors, achieving near state-of-the-art on a wide range of phrase similarity tasks and relation classification; meanwhile, DCS can guide building vectors for structured queries that can be directly executed. We evaluate this utility on sentence completion task and report a new state-of-the-art.

1 Introduction

A major goal of semantic processing is to map natural language utterances to representations that facilitate calculation of meanings, execution of commands, and/or inference of knowledge. Formal semantics supports such representations by defining words as some functional units and combining them via a specific logic. A simple and illustrative example is the Dependency-based Compositional Semantics (DCS) (Liang et al., 2013). DCS composes meanings from denotations of words (i.e. sets of things to which the words apply); say, the denotations of the concept `drug` and the event `ban` is shown in Figure 1b, where `drug` is a list of drug names and `ban` is a list of the subject-complement pairs in any *ban* event; then, a list of *banned drugs* can be constructed by first taking the COMP column of all records in `ban` (projection "π_{COMP}"), and then intersecting the results with `drug` (intersection "$\cap$"). This procedure defined how words can be combined to form a meaning. Better yet, the procedure can be concisely illustrated by the DCS tree of "*banned drugs*" (Figure 1a), which is similar to a dependency tree but possesses precise procedural and logical meaning (Section 2). DCS has been shown useful in question answering (Liang et al., 2013) and textual entailment recognition (Tian et al., 2014).

Orthogonal to the formal semantics of DCS, distributional vector representations are useful in capturing lexical semantics of words (Turney and Pantel, 2010; Levy et al., 2015), and progress is made in combining the word vectors to form meanings of phrases/sentences (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012; Paperno et al., 2014; Hashimoto et al., 2014). However, less effort is devoted to finding a link between vector-based compositions and the composition operations in any formal semantics. We believe that if a link can be found, then symbolic formulas in the formal semantics will be realized by vectors composed from word embeddings, such that similar things are realized by similar vectors; meanwhile, vectors will acquire formal meanings that can directly be used in execution or inference process. Still, to find a link is challenging because any vector compositions that realize such a link must conform to the logic of the formal semantics.

In this paper, we establish a link between DCS and certain vector compositions, achieving a *vector-based DCS* by replacing denotations of words with word vectors, and realizing the composition operations such as intersection and projection as addition and linear mapping, respectively. For example, to construct a vector for "*banned drugs*", one takes the word vector $\mathbf{v}_{ban}$ and multiply it by a matrix M_{COMP}, corresponding to the projection π_{COMP}; then, one adds the result to the word vector $\mathbf{v}_{drug}$ to realize the intersection operation (Figure 1c). We provide a method to train the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1277–1287,
Berlin, Germany, August 7-12, 2016.

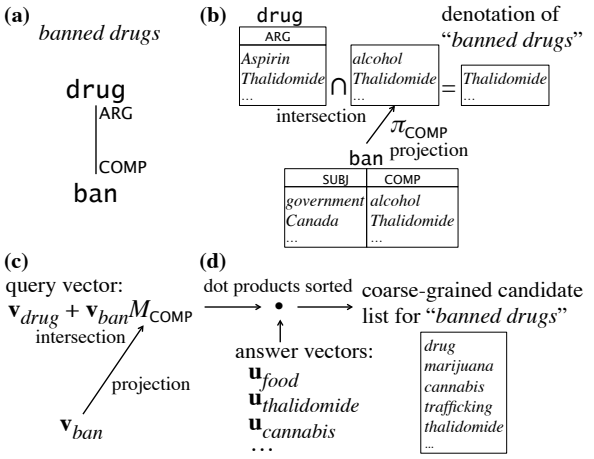

Figure 1: (a) The DCS tree of "*banned drugs*", which controls (b) the calculation of its denotation. In this paper, we learn word vectors and matrices such that (c) the same calculation is realized in distributional semantics. The constructed query vector can be used to (d) retrieve a list of coarse-grained candidate answers to that query.

word vectors and linear mappings (i.e. matrices) jointly from unlabeled corpora.

The rationale for our model is as follows. First, recent research has shown that additive composition of word vectors is an approximation to the situation where two words have overlapping context (Tian et al., 2015); therefore, it is suitable to implement an "and" or intersection operation (Section 3). We design our model such that the resulted distributional representations are expected to have additive compositionality. Second, when intersection is realized as addition, it is natural to implement projection as linear mapping, as suggested by the logical interactions between the two operations (Section 3). Experimentally, we show that vectors and matrices learned by our model exhibit favorable characteristics as compared with vectors trained by GloVe (Pennington et al., 2014) or those learned from syntactic dependencies (Section 5.1). Finally, additive composition brings our model a strong ability to calculate similar vectors for similar phrases, whereas syntactic-semantic roles (e.g. SUBJ, COMP) can be distinguished by different projection matrices (e.g. M_{SUBJ}, M_{COMP}). We achieve near state-of-the-art performance on a wide range of phrase similarity tasks (Section 5.2) and relation classification (Section 5.3).

Furthermore, we show that a vector as constructed above for "*banned drugs*" can be used as a *query vector* to retrieve a coarse-grained candidate list of banned drugs, by sorting its dot products with *answer vectors* that are also learned by our model (Figure 1d). This is due to the ability of our approach to provide a language model that can find likely words to fill in the blanks such as "__ *is a banned drug*" or "*the drug* __ *is banned by* ...". A highlight is the calculation being done as if a query is "executed" by the DCS tree of "*banned drugs*". We quantitatively evaluate this utility on sentence completion task (Zweig et al., 2012) and report a new state-of-the-art (Section 5.4).

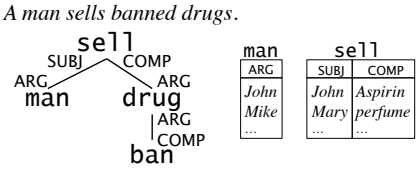

Figure 2: DCS tree for a sentence

2 DCS Trees

DCS composes meanings from denotations, or sets of things to which words apply. A "thing" (i.e. element of a denotation) is represented by a tuple of features of the form Field=*Value*, with a fixed inventory of fields. For example, a denotation `ban` might be a set of tuples $\texttt{ban} = \{(\mathsf{SUBJ}{=}\textit{Canada}, \mathsf{COMP}{=}\textit{Thalidomide}), \ldots\}$, in which each tuple records participants of a banning event (e.g. Canada banning Thalidomide).

Operations are applied to sets of things to generate new denotations, for modeling semantic composition. An example is the intersection of `pet` and `fish` giving the denotation of "*pet fish*". Another necessary operation is projection; by π_{N} we mean a function mapping a tuple to its value of the field N. For example, $\pi_{\mathsf{COMP}}(\texttt{ban})$ is the value set of the COMP fields in `ban`, which consists of banned objects (i.e. $\{\textit{Thalidomide}, \ldots\}$). In this paper, we assume a field ARG to be names of things representing themselves, hence for example $\pi_{\mathsf{ARG}}(\texttt{drug})$ is the set of names of drugs.

For a value set V, we also consider inverse image $\pi_{\mathsf{N}}^{-1}(V) := \{x \mid \pi_{\mathsf{N}}(x) \in V\}$. For example,

$$\texttt{D}_1 := \pi_{\mathsf{SUBJ}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{man}))$$

consists of all tuples of the form $(\mathsf{SUBJ}{=}x, \ldots)$, where x is a man's name (i.e. $x \in \pi_{\mathsf{ARG}}(\texttt{man})$). Thus, $\texttt{sell} \cap \texttt{D}_1$ denotes men's selling events (i.e. $\{(\mathsf{SUBJ}{=}\textit{John}, \mathsf{COMP}{=}\textit{Aspirin}), \ldots\}$ as in Figure 2). Similarly, the denotation of "*banned*

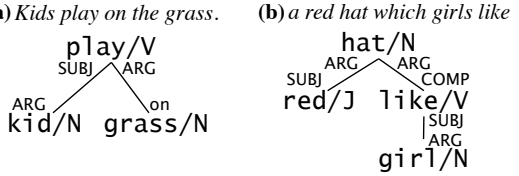

Figure 3: DCS trees in this work

drugs" as in Figure 1b is formally written as

$$\mathtt{D}_2 := \mathtt{drug} \cap \pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{COMP}}(\mathtt{ban})),$$

Hence the following denotation

$$\mathtt{D}_3 := \mathtt{sell} \cap \mathtt{D}_1 \cap \pi^{-1}_{\mathsf{COMP}}(\pi_{\mathsf{ARG}}(\mathtt{D}_2))$$

consists of selling events such that the SUBJ is a man and the COMP is a banned drug.

The calculation above can proceed in a recursive manner controlled by DCS trees. The DCS tree for the sentence "*a man sells banned drugs*" is shown in Figure 2. Formally, a DCS tree is defined as a rooted tree in which nodes are denotations of content words and edges are labeled by fields at each ends. Assume a node $\mathtt{x}$ has children $\mathtt{y}_1, \ldots, \mathtt{y}_n$, and the edges $(\mathtt{x}, \mathtt{y}_1), \ldots, (\mathtt{x}, \mathtt{y}_n)$ are labeled by $(\mathsf{P}_1, \mathsf{L}_1), \ldots, (\mathsf{P}_n, \mathsf{L}_n)$, respectively. Then, the denotation $[\![\mathtt{x}]\!]$ of the subtree rooted at $\mathtt{x}$ is recursively calculated as

$$[\![\mathtt{x}]\!] := \mathtt{x} \cap \bigcap_{i=1}^{n} \pi^{-1}_{\mathsf{P}_i}(\pi_{\mathsf{L}_i}([\![\mathtt{y}_i]\!])). \qquad (1)$$

As a result, the denotation of the DCS tree in Figure 2 is the denotation $\mathtt{D}_3$ of "*a man sells banned drugs*" as calculated above. DCS can be further extended to handle phenomena such as quantifiers or superlatives (Liang et al., 2013; Tian et al., 2014). In this paper, we focus on the basic version, but note that it is already expressive enough to at least partially capture the meanings of a large portion of phrases and sentences.

DCS trees can be learned from question-answer pairs and a given database of denotations (Liang et al., 2013), or they can be extracted from dependency trees if no database is specified, by taking advantage of the observation that DCS trees are similar to dependency trees (Tian et al., 2014). We use the latter approach, obtaining DCS trees by rule-based conversion from universal dependency (UD) trees (McDonald et al., 2013). Therefore, nodes in a DCS tree are content words in a UD tree, which are in the form of lemma-POS pairs (Figure 3). The inventory of fields is designed to be ARG, SUBJ, COMP, and all prepositions. Prepositions are unlike content words which denote sets of things, but act as relations which we treat similarly as SUBJ and COMP. For example, a prepositional phrase attached to a verb (e.g. *play on the grass*) is treated as in Figure 3a. The presence of two field labels on each edge of a DCS tree makes it convenient for modeling semantics in several cases, such as a relative clause (Figure 3b).

3 Vector-based DCS

For any content word w, we use a query vector $\mathbf{v}_w$ to model its denotation, and an answer vector $\mathbf{u}_w$ to model a prototypical element in that denotation. Query vector $\mathbf{v}$ and answer vector $\mathbf{u}$ are learned such that $\exp(\mathbf{v} \cdot \mathbf{u})$ is proportional to the probability of $\mathbf{u}$ answering the query $\mathbf{v}$. The learning source is a collection of DCS trees, based on the idea that the DCS tree of a declarative sentence usually has non-empty denotation. For example, "*kids play*" means there exists some kid who plays. Consequently, some element in the play denotation belongs to $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$, and some element in the kid denotation belongs to $\pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{SUBJ}}(\mathtt{play}))$. This is a signal to increase the dot product of $\mathbf{u}_{play}$ and the query vector of $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$, as well as the dot product of $\mathbf{u}_{kid}$ and the query vector of $\pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{SUBJ}}(\mathtt{play}))$. When optimized on a large corpus, the "typical" elements of play and kid should be learned by $\mathbf{u}_{play}$ and $\mathbf{u}_{kid}$, respectively. In general, one has

Theorem 1 *Assume the denotation of a DCS tree is not empty. Given any path from node* $\mathtt{x}$ *to* $\mathtt{y}$*, assume edges along the path are labeled by* $(\mathsf{P}, \mathsf{L}), \ldots, (\mathsf{K}, \mathsf{N})$*. Then, an element in the denotation* $\mathtt{y}$ *belongs to* $\pi^{-1}_{\mathsf{N}}(\pi_{\mathsf{K}}(\ldots(\pi^{-1}_{\mathsf{L}}(\pi_{\mathsf{P}}(\mathtt{x}) \ldots).$

Therefore, for any two nodes in a DCS tree, the path from one to another forms a training example, which signals increasing the dot product of the corresponding query and answer vectors.

It is noteworthy that the above formalization happens to be closely related to the skip-gram model (Mikolov et al., 2013b). The skip-gram learns a target vector $\mathbf{v}_w$ and a context vector $\mathbf{u}_w$ for each word w. It assumes the probability of a word y co-occurring with a word x in a context window is proportional to $\exp(\mathbf{v}_x \cdot \mathbf{u}_y)$. Hence, if x and y co-occur within a context window, then one gets a signal to increase $\mathbf{v}_x \cdot \mathbf{u}_y$. If the context window is taken as the same DCS tree, then

the learning of skip-gram and vector-based DCS will be almost the same, except that the target vector $\mathbf{v}_x$ becomes the query vector $\mathbf{v}$, which is no longer assigned to the word x but the path from x to y in the DCS tree (e.g. the query vector for $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$ instead of $\mathbf{v}_{kid}$). Therefore, our model can also be regarded as extending skip-gram to take account of the changes of meanings caused by different syntactic-semantic roles.

Additive Composition Word vectors trained by skip-gram are known to be semantically additive, such as exhibited in word analogy tasks. An effect of adding up two skip-gram vectors is further analyzed in Tian et al. (2015). Namely, the target vector $\mathbf{v}_w$ can be regarded as encoding the distribution of context words surrounding w. If another word x is given, $\mathbf{v}_w$ can be decomposed into two parts, one encodes context words shared with x, and another encodes context words not shared. When $\mathbf{v}_w$ and $\mathbf{v}_x$ are added up, the non-shared part of each of them tend to cancel out, because non-shared parts have nearly independent distributions. As a result, the shared part gets reinforced. An error bound is derived to estimate how close $\frac{1}{2}(\mathbf{v}_w + \mathbf{v}_x)$ gets to the distribution of the shared part. We can see the same mechanism exists in vector-based DCS. In a DCS tree, two paths share a context word if they lead to a same node y; semantically, this means some element in the denotation $\mathtt{y}$ belongs to both denotations of the two paths (e.g. given the sentence "*kids play balls*", $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$ and $\pi^{-1}_{\mathsf{COMP}}(\pi_{\mathsf{ARG}}(\mathtt{ball}))$ both contain a playing event whose SUBJ is a kid and COMP is a ball). Therefore, addition of query vectors of two paths approximates their intersection because the shared context y gets reinforced.

Projection Generally, for any two denotations $\mathtt{X}_1, \mathtt{X}_2$ and any projection π_{N}, we have

$$\pi_{\mathsf{N}}(\mathtt{X}_1 \cap \mathtt{X}_2) \subseteq \pi_{\mathsf{N}}(\mathtt{X}_1) \cap \pi_{\mathsf{N}}(\mathtt{X}_2). \quad (2)$$

And the "$\subseteq$" can often become "$=$", for example when π_{N} is a one-to-one map or $\mathtt{X}_1 = \pi^{-1}_{\mathsf{N}}(V)$ for some value set V. Therefore, if intersection is realized by addition, it will be natural to realize projection by linear mapping because

$$(\mathbf{v}_1 + \mathbf{v}_2)M_{\mathsf{N}} = \mathbf{v}_1 M_{\mathsf{N}} + \mathbf{v}_2 M_{\mathsf{N}} \quad (3)$$

holds for any vectors $\mathbf{v}_1, \mathbf{v}_2$ and any matrix M_{N}, which is parallel to (2). If π_{N} is realized by a matrix M_{N}, then π^{-1}_{N} should correspond to the inverse matrix M^{-1}_{N}, because $\pi_{\mathsf{N}}(\pi^{-1}_{\mathsf{N}}(V)) = V$ for any value set V. So we have realized all composition operations in DCS.

Query vector of a DCS tree Now, we can define the query vector of a DCS tree as parallel to (1):

$$\mathbf{v}_{[\![\mathtt{x}]\!]} := \mathbf{v}_x + \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_{[\![\mathtt{y}_i]\!]} M_{\mathsf{L}_i} M^{-1}_{\mathsf{P}_i}. \quad (4)$$

4 Training

As described in Section 3, vector-based DCS assigns a query vector $\mathbf{v}_w$ and an answer vector $\mathbf{u}_w$ to each content word w. And for each field N, it assigns two matrices M_{N} and M^{-1}_{N}. For any path from node x to y sampled from a DCS tree, assume the edges along are labeled by $(\mathsf{P}, \mathsf{L}), \ldots, (\mathsf{K}, \mathsf{N})$. Then, the dot product $\mathbf{v}_x M_{\mathsf{P}} M^{-1}_{\mathsf{L}} \ldots M_{\mathsf{K}} M^{-1}_{\mathsf{N}} \cdot \mathbf{u}_y$ gets a signal to increase.

Formally, we adopt the noise-contrastive estimation (Gutmann and Hyvärinen, 2012) as used in the skip-gram model, and mix the paths sampled from DCS trees with artificially generated noise. Then, $\sigma(\mathbf{v}_x M_{\mathsf{P}} M^{-1}_{\mathsf{L}} \ldots M_{\mathsf{K}} M^{-1}_{\mathsf{N}} \cdot \mathbf{u}_y)$ models the probability of a training example coming from DCS trees, where $\sigma(\theta) = 1/\{1 + \exp(-\theta)\}$ is the sigmoid function. The vectors and matrices are trained by maximizing the log-likelihood of the mixed data. We use stochastic gradient descent (Bottou, 2012) for training. Some important settings are discussed below.

Noise For any $\mathbf{v}_x M_1 M_2^{-1} \ldots M_{2l-1} M_{2l}^{-1} \cdot \mathbf{u}_y$ obtained from a path of a DCS tree, we generate noise by randomly choosing an index $i \in [2, 2l]$, and then replacing M_j or M_j^{-1} ($\forall j \geq i$) and $\mathbf{u}_y$ by $M_{\mathsf{N}(j)}$ or $M^{-1}_{\mathsf{N}(j)}$ and $\mathbf{u}_z$, respectively, where $\mathsf{N}(j)$ and z are independently drawn from the marginal (i.e. unigram) distributions of fields and words.

Update For each data point, when i is the chosen index above for generating noise, we view indices $j < i$ as the "target" part, and $j >= i$ as the "context", which is completely replaced by the noise, as an analogous to the skip-gram model. Then, at each step we only update one vector and one matrix from each of the target, context, and noise part; more specifically, we only update $\mathbf{v}_x$, M_{i-1} or M^{-1}_{i-1}, M_i or M_i^{-1}, $M_{\mathsf{N}(i)}$ or $M^{-1}_{\mathsf{N}(i)}$, $\mathbf{u}_y$ and $\mathbf{u}_z$, at the step. This is much faster than always updating all matrices.

Initialization Matrices are initialized as $\frac{1}{2}(I + G)$, where I is the identity matrix; and G and all

GloVe	no matrix	vecDCS	vecUD
books	*essay/N*	*novel/N*	*essay/N*
author	*novel/N*	*essay/N*	*novel/N*
published	*memoir/N*	*anthology/N*	*article/N*
novel	*books/N*	*publication/N*	*anthology/N*
memoir	*autobiography/N*	*memoir/N*	*poem/N*
wrote	***non-fiction/J***	*poem/N*	*autobiography/N*
biography	***reprint/V***	*autobiography/N*	*publication/N*
autobiography	***publish/V***	*story/N*	*journal/N*
essay	***republish/V***	*pamphlet/N*	*memoir/N*
illustrated	*chapbook/N*	*tale/N*	*pamphlet/N*

Table 1: Top 10 similar words to "*book/N*"

vectors are initialized with i.i.d. Gaussians of variance $1/d$, where d is the vector dimension. We find that the diagonal component I is necessary to bring information from $\mathbf{v}_x$ to $\mathbf{u}_y$, whereas the randomness of G makes convergence faster. M_{N}^{-1} is initialized as the transpose of M_{N}.

Learning Rate We find that the initial learning rate for vectors can be set to 0.1. But for matrices, it should be less than 0.0005 otherwise the model diverges. For stable training, we rescale gradients when their norms exceed a threshold.

Regularizer During training, M_{N} and M_{N}^{-1} are treated as independent matrices. However, we use the regularizer $\gamma\|M_{\mathsf{N}}^{-1}M_{\mathsf{N}} - \frac{1}{d}\operatorname{tr}(M_{\mathsf{N}}^{-1}M_{\mathsf{N}})I\|^2$ to drive M_{N}^{-1} close to the inverse of M_{N}.[1] We also use $\kappa\|M_{\mathsf{N}}^{\perp}M_{\mathsf{N}} - \frac{1}{d}\operatorname{tr}(M_{\mathsf{N}}^{\perp}M_{\mathsf{N}})I\|^2$ to prevent M_{N} from having too different scales at different directions (i.e., to drive M_{N} close to orthogonal). We set $\gamma = 0.001$ and $\kappa = 0.0001$. Despite the rather weak regularizer, we find that M_{N}^{-1} can be learned to be exactly the inverse of M_{N}, and M_{N} can actually be an orthogonal matrix, showing some semantic regularity (Section 5.1).

5 Experiments

For training vector-based DCS, we use Wikipedia Extractor[2] to extract texts from the 2015-12-01 dump of English Wikipedia[3]. Then, we use Stanford Parser[4] (Klein and Manning, 2003) to parse all sentences and convert the UD trees into DCS trees by handwritten rules. We assign a weight to each path of the DCS trees as follows.

[1]Problem with the naive regularizer $\|M^{-1}M - I\|^2$ is that, when the scale of M goes larger, it will drive M^{-1} smaller, which may lead to degeneration. So we scale I according to the trace of $M^{-1}M$.

[2]`http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[3]`https://dumps.wikimedia.org/enwiki/`

[4]`http://nlp.stanford.edu/software/lex-parser.shtml`

$\pi_{\mathsf{SUBJ}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{house}))$	$\pi_{\mathsf{COMP}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{house}))$	$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{in}}(\texttt{house}))$
victorian/J	*build/V*	*sit/V*
stand/V	*rent/V*	*house/N*
vacant/J	*leave/V*	*stand/V*
18th-century/J	*burn down/V*	*live/V*
historic/J	*remodel/V*	*hang/V*
old/J	*demolish/V*	*seat/N*
georgian/J	*restore/V*	*stay/V*
local/J	*renovate/V*	*serve/V*
19th-century/J	*rebuild/V*	*reside/V*
tenement/J	*construct/V*	*hold/V*
$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{SUBJ}}(\texttt{learn}))$	$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{COMP}}(\texttt{learn}))$	$\pi_{\mathsf{about}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{learn}))$
teacher/N	*skill/N*	*otherness/N*
skill/N	*lesson/N*	*intimacy/N*
he/P	*technique/N*	*femininity/N*
she/P	*experience/N*	*self-awareness/N*
therapist/N	*ability/N*	*life/N*
student/N	*something/N*	*self-expression/N*
they/P	*knowledge/N*	*sadomasochism/N*
mother/N	*language/N*	*emptiness/N*
lesson/N	*opportunity/N*	*criminality/N*
father/N	*instruction/N*	*masculinity/N*

Table 2: Top 10 answers of high dot products

For any path P passing through k intermediate nodes of degrees $n_1, \ldots, n_k$, respectively, we set

$$\mathrm{Weight}(P) := \prod_{i=1}^{k} \frac{1}{n_i - 1}. \qquad (5)$$

Note that $n_i \geq 2$ because there is a path P passing through the node; and $\mathrm{Weight}(P) = 1$ if P consists of a single edge. The equation (5) is intended to degrade long paths which pass through several high-valency nodes. We use a random walk algorithm to sample paths such that the expected times a path is sampled equals its weight. As a result, the sampled path lengths range from 1 to 19, average 2.1, with an exponential tail. We convert all words which are sampled less than 1000 times to `*UNKNOWN*/POS`, and all prepositions occurring less than 10000 times to an *UNKNOWN* field. As a result, we obtain a vocabulary of 109k words and 211 field names.

Using the sampled paths, vectors and matrices are trained as in Section 4 (vecDCS). The vector dimension is set to $d = 250$. We compare with three baselines: (i) all matrices are fixed to identity ("no matrix"), in order to investigate the effects of meaning changes caused by syntactic-semantic roles and prepositions; (ii) the regularizer enforcing M_{N}^{-1} to be actually the inverse matrix of M_{N} is set to $\gamma = 0$ ("no inverse"), in order to investigate the effects of a semantically motivated constraint; and (iii) applying the same training scheme to UD trees directly, by modeling UD relations as matrices ("vecUD"). In this case, one edge is assigned one UD relation rel, so we implement the transfor-

	AN	NN	VO	SVO	GS11	GS12
vecDCS	**0.51**	**0.49**	**0.41**	**0.62**	0.29	0.33
-no matrix	**0.52**	0.46	**0.42**	**0.62**	0.29	0.33
-no inverse	0.47	0.43	0.38	0.58	0.28	0.33
vecUD	0.44	0.46	**0.41**	0.58	0.25	0.25
GloVe	0.41	0.47	**0.41**	0.60	0.23	0.17
Grefenstette and Sadrzadeh (2011)	-	-	-	-	0.21	-
Blacoe and Lapata (2012):RAE	0.31	0.30	0.28	-	-	-
Grefenstette (2013a)	-	-	-	-	-	0.27
Paperno et al. (2014)	-	-	-	-	-	**0.36**
Hashimoto et al. (2014):Wadd_{nl}	0.48	0.40	0.39		0.34	-
Kartsaklis and Sadrzadeh (2014)	-	-	-	0.43	**0.41**	-

Table 3: Spearman's ρ on phrase similarity

mation from child to parent by M_{rel}, and from parent to child by M_{rel}^{-1}. The same hyper-parameters are used to train vecUD. By comparing vecDCS with vecUD we investigate if applying the semantics framework of DCS makes any difference. Additionally, we compare with the GloVe (6B, 300d) vector[5] (Pennington et al., 2014). Norms of all word vectors are normalized to 1 and Frobenius norms of all matrices are normalized to $\sqrt{d}$.

5.1 Qualitative Analysis

We observe several special properties of the vectors and matrices trained by our model.

Words are clustered by POS In terms of cosine similarity, word vectors trained by vecDCS and vecUD are clustered by POS tags, probably due to their interactions with matrices during training. This is in contrast to the vectors trained by GloVe or "no matrix" (Table 1).

Matrices show semantic regularity Matrices learned for ARG, SUBJ and COMP are exactly orthogonal, and some most frequent prepositions[6] are remarkably close. For these matrices, the corresponding M^{-1} also exactly converge to their inverse. It suggests regularities in the semantic space, especially because orthogonal matrices preserve cosine similarity – if M_{N} is orthogonal, two words x, y and their projections $\pi_{\mathsf{N}}(\mathtt{x})$, $\pi_{\mathsf{N}}(\mathtt{y})$ will have the same similarity measure, which is semantically reasonable. In contrast, matrices trained by vecUD are only orthogonal for three UD relations, namely conj, dep and appos.

Words transformed by matrices To illustrate the matrices trained by vecDCS, we start from the query vectors of two words, house and learn, applying different matrices to them, and show the 10 answer vectors of the highest dot products (Tabel 2). These are the lists of likely words which: take *house* as a subject, take *house* as a complement, fills into "__ *in house*", serve as a subject of *learn*, serve as a complement of *learn*, and fills into "*learn about* __", respectively. As the table shows, matrices in vecDCS are appropriately learned to map word vectors to their syntactic-semantic roles.

5.2 Phrase Similarity

To test if vecDCS has the composition ability to calculate similar things as similar vectors, we conduct evaluation on a wide range of phrase similarity tasks. In these tasks, a system calculates similarity scores for pairs of phrases, and the performance is evaluated as its correlation with human annotators, measured by Spearman's ρ.

Datasets Mitchell and Lapata (2010) create datasets[7] for pairs of three types of two-word phrases: adjective-nouns (AN) (e.g. "*black hair*" and "*dark eye*"), compound nouns (NN) (e.g. "*tax charge*" and "*interest rate*") and verb-objects (VO) (e.g. "*fight war*" and "*win battle*"). Each dataset consists of 108 pairs and each pair is annotated by 18 humans (i.e., 1,944 scores in total). Similarity scores are integers ranging from 1 to 7. Another dataset[8] is created by extending VO to Subject-Verb-Object (SVO), and then assessing similarities by crowd sourcing (Kartsaklis and Sadrzadeh, 2014). The dataset GS11 created by Grefenstette and Sadrzadeh (2011) (100 pairs, 25 annotators) is also of the form SVO, but in each pair only the verbs are different (e.g. "*man pro-*

[5] http://nlp.stanford.edu/projects/glove/

[6] of, in, to, for, with, on, as, at, from

[7] http://homepages.inf.ed.ac.uk/s0453356/

[8] http://www.cs.ox.ac.uk/activities/compdistmeaning/

Message-Topic(e_1, e_2)	*It is a monthly* [*report*]$_1$ *providing* [*opinion*]$_2$ *and advice on current United States government contract issues.*
Message-Topic(e_1, e_2)	*The* [*report*]$_1$ *gives an account of the silvicultural* [*work*]$_2$ *done in Africa, Asia, Australia, South American and the Caribbean.*
Message-Topic(e_1, e_2)	*NUS today responded to the Government's* [*announcement*]$_1$ *of the long-awaited* [*review*]$_2$ *of university funding.*
Component-Whole(e_2, e_1)	*The* [*review*]$_1$ *published political* [*commentary*]$_2$ *and opinion, but even more than that.*
Message-Topic(e_1, e_2)	*It is a 2004* [*book*]$_1$ *criticizing the political and linguistic* [*writings*]$_2$ *of Noam Chomsky.*

Table 4: Similar training instances clustered by cosine similarities between features

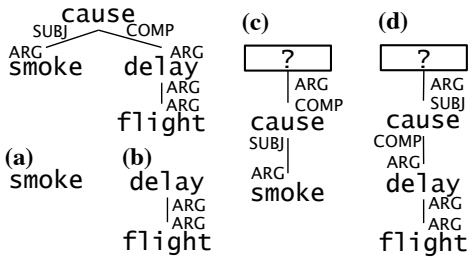

Figure 4: For "[*smoke*]$_1$ *cause flight* [*delay*]$_2$", we construct (a)(b) from subtrees, and (c)(d) from re-rooted trees, to form 4 query vectors as feature.

vide/supply money"). The dataset GS12 described in Grefenstette (2013a) (194 pairs, 50 annotators) is of the form Adjective-Noun-Verb-Adjective-Noun (e.g. "*local family run/move small hotel*"), where only verbs are different in each pair.

Our method We calculate the cosine similarity of query vectors corresponding to phrases. For example, the query vector for "*fight war*" is calculated as $\mathbf{v}_{war} M_{\mathsf{ARG}} M_{\mathsf{COMP}}^{-1} + \mathbf{v}_{fight}$. For vecUD we use M_{nsubj} and M_{dobj} instead of M_{SUBJ} and M_{COMP}, respectively. For GloVe we use additive compositions.

Results As shown in Table 3, vecDCS is competitive on AN, NN, VO, SVO and GS12, consistently outperforming "no inverse", vecUD and GloVe, showing strong compositionality. The weakness of "no inverse" suggests that relaxing the constraint of inverse matrices may hurt compositionaly, though our preliminary examination on word similarities did not find any difference. The GS11 dataset appears to favor models that can learn from interactions between the subject and object arguments, such as the non-linear model Wadd_{nl} in Hashimoto et al. (2014) and the entanglement model in Kartsaklis and Sadrzadeh (2014). However, these models do not show particular advantages on other datasets. The recursive autoencoder (RAE) proposed in Socher et al. (2011) shares an aspect with vecDCS as to construct meanings from parse trees. It is tested by Blacoe and Lapata (2012) for compositionality, where vecDCS appears to be better. Nevertheless, we note that "no matrix" performs as good as vecDCS, suggesting that meaning changes caused by syntactic-semantic roles might not be major factors in these datasets, because the syntactic-semantic relations are all fixed in each dataset.

vecDCS	81.2
-no matrix	69.2
-no inverse	79.7
vecUD	69.2
GloVe	74.1
Socher et al. (2012)	79.1
+3 features	82.4
dos Santos et al. (2015)	84.1
Xu et al. (2015)	**85.6**

Table 5: F1 on relation classification

5.3 Relation Classification

In a relation classification task, the relation between two words in a sentence needs to be classified; we expect vecDCS to perform better than "no matrix" on this task because vecDCS can distinguish the different syntactic-semantic roles of the two slots the two words fit in. We confirm this conjecture in this section.

Dataset We use the dataset of SemEval-2010 Task 8 (Hendrickx et al., 2009), in which 9 directed relations (e.g. *Cause-Effect*) and 1 undirected relation *Other* are annotated, 8,000 instances for training and 2,717 for test. Performance is measured by the 9-class direction-aware Macro-F1 score excluding *Other* class.

Our method For any sentence with two words marked as e_1 and e_2, we construct the DCS tree of the sentence, and take the subtree T rooted at the common ancestor of e_1 and e_2. We construct four vectors from T, namely: the query vector for the subtree rooted at e_1 (resp. e_2), and the query vector of the DCS tree obtained from T by re-rooting it at e_1 (resp. e_2) (Figure 4). The four vectors are normalized and concatenated to form the only feature used to train a classifier. For vecUD, we use the corresponding vectors calculated from UD trees. For GloVe, we use the word vector of e_1 (resp. e_2), and the sum of vectors of all words within the span $[e_1, e_2)$ (resp. $(e_1, e_2]$) as

"banned drugs"	*"banned movies"*	*"banned books"*
drug/N	*bratz/N*	*publish/N*
marijuana/N	*porn/N*	*unfair/N*
cannabis/N	*indecent/N*	*obscene/N*
trafficking/N	*blockbuster/N*	*samizdat/N*
thalidomide/N	*movie/N*	*book/N*
smoking/N	*idiots/N*	*responsum/N*
narcotic/N	*blacklist/N*	*illegal/N*
botox/N	*grindhouse/N*	*reclaiming/N*
doping/N	*doraemon/N*	*redbook/N*

Table 6: Answers for composed query vectors

vecDCS	50
-no matrix	**60**
-no inverse	46
vecUD	31
N-gram (Various)	39-41
Zweig et al. (2012)	52
Mnih and Teh (2012)	55
Gubbins and Vlachos (2013)	50
Mikolov et al. (2013a)	55

Table 7: Accuracy (%) on sentence completion

the four vectors. Classifier is SVM[9] with RBF kernel, $C = 2$ and $\Gamma = 0.25$. The hyper-parameters are selected by 5-fold cross validation.

Results VecDCS outperforms baselines on relation classification (Table 5). It makes 16 errors in misclassifying the direction of a relation, as compared to 144 such errors made by "no matrix", 23 by "no inverse", 30 by vecUD, and 161 by GloVe. This suggests that models with syntactic-semantic transformations (i.e. vecDCS, "no inverse", and vecUD) are indeed good at distinguishing the different roles played by e_1 and e_2. VecDCS scores moderately lower than the state-of-the-art (Xu et al., 2015), however we note that these results are achieved by adding additional features and training task-specific neural networks (dos Santos et al., 2015; Xu et al., 2015). Our method only uses features constructed from unlabeled corpora. From this point of view, it is comparable to the MV-RNN model (without features) in Socher et al. (2012), and vecDCS actually does better. Table 4 shows an example of clustered training instances as assessed by cosine similarities between their features. It suggests that the features used in our method can actually cluster similar relations.

5.4 Sentence Completion

If vecDCS can compose query vectors of DCS trees, one should be able to "execute" the vectors to get a set of answers, as the original DCS trees can do. This is done by taking dot products with answer vectors and then ranking the answers. Examples are shown in Table 6. Since query vectors and answer vectors are trained from unlabeled corpora, we can only obtain a coarse-grained candidate list. However, it is noteworthy that despite a common word "*banned*" shared by the phrases, their answer lists are largely different, suggesting that composition actually can be done. Moreover, some words indeed answer the queries (e.g. *Thalidomide* for "banned drugs" and *Samizdat* for "*banned books*").

Quantitatively, we evaluate this utility of executing queries on the sentence completion task. In this task, a sentence is presented with a blank that need to be filled in. Five possible words are given as options for each blank, and a system needs to choose the correct one. The task can be viewed as a coarse-grained question answering or an evaluation for language models (Zweig et al., 2012). We use the MSR sentence completion dataset[10] which consists of 1,040 test questions and a corpus for training language models. We train vecDCS on this corpus and use it for evaluation.

Results As shown in Table 7, vecDCS scores better than the N-gram model and demonstrates promising performance. However, to our surprise, "no matrix" shows an even better result which is the new state-of-the-art. Here we might be facing the same problem as in the phrase similarity task (Section 5.2); namely, all choices in a question fill into the same blank and the same syntactic-semantic role, so the transforming matrices in vecDCS might not be able to distinguish different choices; on the other hand, vecDCS would suffer more from parsing and POS-tagging errors. Nonetheless, we believe the result by "no matrix" reveals a new horizon of sentence completion, and suggests that composing semantic vectors according to DCS trees could be a promising direction.

6 Discussion

We have demonstrated a way to link a vector composition model to a formal semantics, combining the strength of vector representations to calculate phrase similarities, and the strength of formal semantics to build up structured queries. In this section, we discuss several lines of previous research related to this work.

[9] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[10] http://research.microsoft.com/en-us/projects/scc/

Logic and Distributional Semantics Logic is necessary for implementing the functional aspects of meaning and organizing knowledge in a structured and unambiguous way. In contrast, distributional semantics provides an elegant methodology for assessing semantic similarity and is well suited for learning from data. There have been repeated calls for combining the strength of these two approaches (Coecke et al., 2010; Baroni et al., 2014; Liang and Potts, 2015), and several systems (Lewis and Steedman, 2013; Beltagy et al., 2014; Tian et al., 2014) have contributed to this direction. In the remarkable work by Beltagy et al. (to appear), word and phrase similarities are explicitly transformed to weighted logical rules that are used in a probabilistic inference framework. However, this approach requires considerable amount of engineering, including the generation of rule candidates (e.g. by aligning sentence fragments), converting distributional similarities to weights, and efficiently handling the rules and inference. What if the distributional representations are equipped with a logical interface, such that the inference can be realized by simple vector calculations? We have shown it possible to realize semantic composition; we believe this may lead to significant simplification of the system design for combining logic and distributional semantics.

Compositional Distributional Models There has been active exploration on how to combine word vectors such that adequate phrase/sentence similarities can be assessed (Mitchell and Lapata, 2010, *inter alia*), and there is nothing new in using matrices to model changes of meanings. However, previous model designs mostly rely on linguistic intuitions (Paperno et al., 2014, *inter alia*), whereas our model has an exact logic interpretation. Furthermore, by using additive composition we enjoy a learning guarantee (Tian et al., 2015).

Vector-based Logic Models This work also shares the spirit with Grefenstette (2013b) and Rocktaeschel et al. (2014), in exploring vector calculations that realize logic operations. However, the previous works did not specify how to integrate contextual distributional information, which is necessary for calculating semantic similarity.

Formal Semantics Our model implements a fragment of logic capable of semantic composition, largely due to the simple framework of Dependency-based Compositional Semantics (Liang et al., 2013). It fits in a long tradition of logic-based semantics (Montague, 1970; Dowty et al., 1981; Kamp and Reyle, 1993), with extensive studies on extracting semantics from syntactic representations such as HPSG (Copestake et al., 2001; Copestake et al., 2005) and CCG (Baldridge and Kruijff, 2002; Bos et al., 2004; Steedman, 2012; Artzi et al., 2015; Mineshima et al., 2015).

Logic for Natural Language Inference The pursue of a logic more suitable for natural language inference is also not new. For example, MacCartney and Manning (2008) has implemented a model of natural logic (Lakoff, 1970). We would not reach the current formalization of logic of DCS without reading the work by Calvanese et al. (1998), which is an elegant formalization of database semantics in description logic.

Semantic Parsing DCS-related representations have been actively used in semantic parsing and we see potential in applying our model. For example, Berant and Liang (2014) convert λ-DCS queries to canonical utterances and assess paraphrases at the surface level; an alternative could be using vector-based DCS to bring distributional similarity directly into calculation of denotations. We also borrow ideas from previous work, for example our training scheme is similar to Guu et al. (2015) in using paths and composition of matrices, and our method is similar to Poon and Domingos (2009) in building structured knowledge from clustering syntactic parse of unlabeled data.

Further Applications Regarding the usability of distributional representations learned by our model, a strong point is that the representation takes into account syntactic/structural information of context. Unlike several previous models (Padó and Lapata, 2007; Levy and Goldberg, 2014; Pham et al., 2015), our approach learns matrices at the same time that can extract the information according to different syntactic-semantic roles. A related application is selectional preference (Baroni and Lenci, 2010; Lenci, 2011; Van de Cruys, 2014), wherein our model might has potential for smoothly handling composition.

Reproducibility Find our code at `https://github.com/tianran/vecdcs`

Acknowledgments This work was supported by CREST, JST. We thank the anonymous reviewers for their valuable comments.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of EMNLP*.

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling ccg and hybrid logic dependency semantics. In *Proceedings of ACL*.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4).

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9(6).

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *Proceedings of ACL*.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. to appear. Representing meaning with a combination of logical form and vectors. *Computational Linguistics*, special issue on formal distributional semantics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of ICCL*.

Léon Bottou. 2012. Stochastic gradient descent tricks. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*. Springer, Berlin.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS98)*.

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

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of ACL*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3).

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL-IJCNLP*.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Springer Netherlands.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*.

Edward Grefenstette. 2013a. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. PhD thesis.

Edward Grefenstette. 2013b. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of *SEM*.

Joseph Gubbins and Andreas Vlachos. 2013. Dependency language models for sentence completion. In *Proceedings of EMNLP*.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1).

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Springer Netherlands.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1-2).

Alessandro Lenci. 2011. Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of ACL*, 3.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of ACL*, 1.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annual Review of Linguistics*, 1.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of Coling*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of EMNLP*.

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

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of ICML*.

Richard Montague. 1970. Universal grammar. *Theoria*, 36.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2).

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of ACL*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of EMNLP*.

Tim Rocktaeschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *ACL Workshop on Semantic Parsing (SP'14)*.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in NIPS*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*.

Mark Steedman. 2012. *Taking Scope - The Natural Semantics of Quantifiers*. MIT Press.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of ACL*.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2015. The mechanism of additive composition. arXiv:1511.08407.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1).

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of EMNLP*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*.

Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of ACL*.