Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5, Takayama, Ikoma, Nara 630-0192, Japan
{masashi-t,kevinduh,shimbo,matsu}@is.naist.jp

Abstract

We present a novel vector space model for semantic co-compositionality. Inspired by Generative Lexicon Theory (Pustejovsky, 1995), our goal is a compositional model where both predicate and argument are allowed to modify each others' meaning representations while generating the overall semantics. This readily addresses some major challenges with current vector space models, notably the polysemy issue and the use of one representation per word type. We implement *co-compositionality* using *prototype projections* on predicates/arguments and show that this is effective in adapting their word representations. We further cast the model as a neural network and propose an unsupervised algorithm to jointly train word representations with co-compositionality. The model achieves the best result to date ($\rho = 0.47$) on the semantic similarity task of transitive verbs (Grefenstette and Sadrzadeh, 2011).

1 Introduction

Vector space models of words have been very successful in capturing the semantic and syntactic characteristics of individual lexical items (Turney and Pantel, 2010). Much research has addressed the question of how to construct individual word representations, for example distributional models (Mitchell and Lapata, 2010) and neural models (Collobert and Weston, 2008). These word representations are used in various natural language processing (NLP) tasks such as part-of-speech tagging, chunking, named entity recognition, and semantic role labeling (Turian et al., 2010; Collobert et al., 2011).

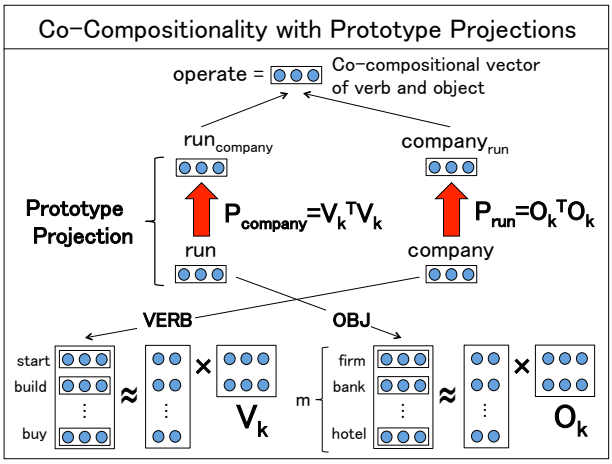

Figure 1: Here, we capture the semantics of *run* in *run company* by projecting the original word representation of *run* to the prototype space of *company* (and vice versa).

Recently, modeling of semantic compositionality (Frege, 1892) in vector space has emerged as another important line of research (Mitchell and Lapata, 2008; Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Socher et al., 2012; Grefenstette and Sadrzadeh, 2011; Van de Cruys et al., 2013). The goal is to formulate how individual word representations ought to be combined to achieve phrasal or sentential semantics.

The main questions for semantic compositionality that we are concerned with are: (1) how can polysemy be handled by a single vector representation per word type, learned by either a distributional or neural model, and (2) how does composition resolve

these ambiguities. To this end, we are inspired by the idea of *type coercion* and *co-compositionality* in Generative Lexicon Theory (Pustejovsky, 1995). Co-compositionality advocates that instead of a predicate-argument view of composition, both predicate and argument influence/coerce each other to generate the overall meaning. For example, consider a polysemous word like *run*:

- (a) He runs the company.
- (b) He runs the marathon.

Run may have several senses, but the prototypical verbs that select for *company* differ from those that select for *marathon*, and thus the ambiguity at the word level is resolved at the sentence level. The same is true for the other direction, where the predicate also coerces meaning to the argument to fit expectation.

We believe that models for semantic composition ought to incorporate elements of co-compositionality. We propose such a model here, using what we call *prototype projections*. For each predicate, we transform its vector representation by projecting it into a latent space that is prototypical of its argument. This projection is performed analogously for each argument as well, and the final meaning is computed by composition of these transformed vectors (Figure 1). In addition, the model is cast as a neural network where word representations could be re-trained or fine-tuned.[1]

Our contributions are two-fold:

1. We propose a novel model for semantic co-compositionality. This model, based on prototype projections, is easy to implement and achieves state-of-the-art performance in the sentence similarity dataset developed by Grefenstette and Sadrzadeh (2011).

2. Our results empirically confirm that existing word representations (eg., SDS and NLM in Section 2) are sufficiently effective at capturing polysemy, as long as we have the proper mechanism to tease out the proper sense during composition. We further propose an unsupervised neural network training algorithm that jointly fine-tunes the word representations within the co-composition model, resulting in even better performance on the sentence similarity task.

We would like to emphasize the second contribution especially. Semantics research is divided in two strands, one focusing on learning word representations without consideration for compositionality, and the other focusing on compositional semantics using the representations only as an input. But issues are actually related from the linguistics perspective, and even more so if we adopt a Generative Lexicon perspective. Our neural network model bridges these two strands of research by modeling co-compositionality and learning word representations simultaneously. We note that methods using context effects have been explored by Erk and Padó (2008; 2009) and Thater et al. (2010; 2011), but to the best of our knowledge, ours is the first model to perform co-compositionality and learning of word representations jointly.

In the following, we first provide background to the word representations employed here (Section 2). We describe the model for co-compositionality in Section 3 and the corresponding neural network in Section 4. Evaluation and experiments are presented in Sections 5 and 6. Finally, we end with related work (Section 7) and conclusions (Section 8).

2 Word Vector Representations

2.1 Simple Distributional Semantic space (SDS) word vectors

Word meaning is often represented in a high dimensional space, where each element corresponds to some contextual element in which the word is found. Mitchell and Lapata (2010) present a co-occurrence-based semantic space called Simple Distributional Semantic space (SDS). Their SDS model uses a context window of five words on either side of the target word and 2,000 vector components, representing the most frequent context words (excluding a list of stop words). These components $v_i(t)$ were set to the ratio of the probability of the context word given the

[1] While we are inspired by co-compositionality, it is important to note that our model does not implement qualia structure and other important components of Generative Lexicon Theory. We operate within the vector space model of distributional semantics, so these ideas are implemented with matrix algebra, which is a natural fit with neural networks.

target word to the probability of the context word overall:

$$v_i(t) = \frac{p(c_i|t)}{p(c_i)} = \frac{freq_{c_i,t} \times freq_{total}}{freq_t \times freq_{c_i}} \quad (1)$$

where $freq_{c_i,t}$, $freq_{total}$, $freq_t$ and $freq_{c_i}$ are the frequencies of the context word c_i with the target word t, the total count of all word tokens, the frequency of the target word t, and the frequency of the context word c_i, respectively.

2.2 Neural Language Model (NLM) word embeddings

Another popular way to learn word representations is based on the Neural Language Model (NLM) (Bengio et al., 2003). In comparison with SDS, NLM tend to be low-dimensional (e.g. 50 dimensions) but employ dense features. These dense feature vectors are usually called word embeddings, and it has been shown that such vectors can capture interesting linear relationships, such as $king - man + woman \approx queen$ (Mikolov et al., 2013). In this work, we adopt the model by Collobert and Weston (2008). The idea is to construct a neural network based on word sequences, where one outputs high scores for n-grams that occur in a large unlabeled corpus and low scores for *nonsense* n-grams where one word is replaced by a random word. This word representation with NLM has been used to good effect, for example in (Turian et al., 2010; Collobert et al., 2011; Huang et al., 2012) where induced word representations are used with sophisticated features to improve performance in various NLP tasks.

Specifically, we first represent the word sequence as a vector $\mathbf{x} = [\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m)]$, where w_i is i_{th} word in the sequence, m is the window size, $\mathbf{d}(w)$ is the vector representation of word w (an n-dimensional column vector) and $[\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m)]$ is the concatenation of word vectors as an input of neural network. Second, we compute the score of the sequence,

$$score(\mathbf{x}) = \mathbf{s}^{\mathrm{T}}(\tanh(\mathbf{W}\mathbf{x} + \mathbf{b})) \quad (2)$$

where $\mathbf{W} \in \mathbb{R}^{h \times (mn)}$ and $\mathbf{s} \in \mathbb{R}^h$ are the first and second layer weights of the neural network, and $\mathbf{b} \in \mathbb{R}^h$ is the bias unit of hidden layer. The superscript T represents transposition, and $\tanh$ is applied element-wise. We also create a corrupted sequence $\mathbf{x}_c = [\mathbf{d}(w_1); \mathbf{d}(w_2); \ldots; \mathbf{d}(w_m{}')]$ where $w_m{}'$ is chosen randomly from the vocabulary. We compute the score of this implicit negative sequence $\mathbf{x}_c$ with the same neural network, $score(\mathbf{x}_c) = \mathbf{s}^{\mathrm{T}}(\tanh(\mathbf{W}\mathbf{x}_c + \mathbf{b}))$. Finally, we get the cost function of this training algorithm as follow.

$$J = \max(0, 1 - score(\mathbf{x}) + score(\mathbf{x}_c)) \quad (3)$$

In order to minimize this cost function, we optimize the parameters $\theta = (\mathbf{s}, \mathbf{W}, \mathbf{b}, \mathbf{x})$ via backpropagation with stochastic gradient descent (SGD).

3 The Model

3.1 Prototype Projection

Generative Lexicon Theory (Pustejovsky, 1995) makes a distinction between accidental polysemy (homonyms, e.g. *bank* as financial institution vs. as river side) and logical polysemy (e.g. figure and ground meanings of *door*). Our model handles both cases using the concept of projection to latent *prototype* space. The fundamental idea is that for each word w and a syntactic/semantic (binary) relation R (such as verb-object relation), w has a set of prototype words with which it frequently occurs in relation R. For example, if w is a word *company*, and R is the object-verb relation, prototype words should include *start*, *build*, and *buy* (Figure 1). For each word-relation pair, we pre-compute the latent semantic subspace spanned by these prototype words.

Later, when we encounter a phrase expressing a relation R between two words w_1 and w_2, each word is first projected onto a latent subspace determined by the other word and relation R. The projection operation shifts the meaning of individual words in accordance with context, and through this operation we realize coercion/co-composition. And finally, the meaning of the phrase is computed from the two projected points in the semantic space.

Let us describe how to compute the latent subspace associated with a word w_0 and a relation R. First, we collect from a corpus a set of prototype words that occur frequently in relation R with target word w_0. So for example in Figure 1, if $w_0 =$

verb	object	landmark	similarity(verb, landmark)	similarity(projected verb, landmark)
run	company	operate	0.40	0.70
meet	criterion	satisfy	0.49	0.71
spell	name	write	0.04	0.50

Table 1: Examples of verb-object pairs. Original verb and landmark verb similarity, prototype projected verb and landmark verb similarity, as measure by cosine using Collobert and Weston's word embeddings. *Meet* has a abstract meaning itself, but after prototype projection with matrix constructed by word vectors of $W(\textit{VerbOf}, \textit{criterion})$, *meet* is more close to meaning of *satisfy*.

company, and R = *VerbOf* is the object-verb relation,

$$W(\textit{VerbOf}, \textit{company}) = \{\textit{start}, \textit{build}, \ldots, \textit{buy}\}.$$

Now let $W(R, w_0) = \{w_1, w_2, \cdots, w_m\}$ be the m prototype words we collected, and let $\mathbf{d}(w)$ denote the n-dimensional (column) vector representation of word w (either by SDS or NLM representation). We make an $m \times n$ matrix $\mathbf{C}_{(R,w_0)}$ by stacking the prototype word vectors, i.e.,

$$\mathbf{C}_{(R,w_0)} = [\mathbf{d}(w_1), \mathbf{d}(w_2), \cdots, \mathbf{d}(w_m)]^{\mathrm{T}} \quad (4)$$

and then apply Singular Value Decomposition (SVD) to extract the latent space from this matrix:

$$\mathbf{C}_{(R,w_0)} \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}}. \quad (5)$$

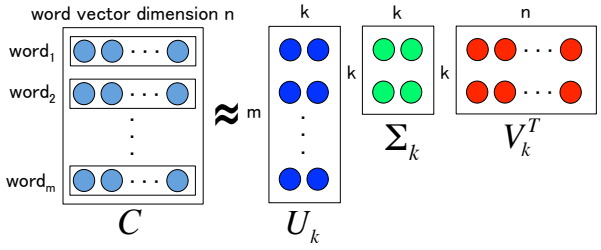

Figure 2: Graphical representation of SVD in our model.

Figure 2 shows the graphical representation of this matrix factorization. In NLP tasks, SVD is often applied to a term-document matrix, but in our model, we apply SVD to the matrix consisting of word vectors.

Intuitively, $\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}}$ represents the latent subspace formed by prototypical words $W(R, w_0) = \{w_1, w_2, \cdots, w_m\}$. We call this matrix the *prototype space* of word w_0 with respect to relation R.

Note that the matrix of orthogonal projection onto this prototype space is given by $\mathbf{P}_{(R,w_0)} = (\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}})^{\mathrm{T}} (\mathbf{\Sigma}_k \mathbf{V}_k^{\mathrm{T}})$. Hence, when we observe a relation $R(w_0, w)$, the projected representation of word w in this context is computed by $\mathbf{prpj}_{(R,w_0)}(w)$ defined as follows:

$$\mathbf{prpj}_{(R,w_0)}(w) = \mathbf{P}_{(R,w_0)} \mathbf{d}(w). \quad (6)$$

Table 1 shows several examples of how meanings change after prototype projection using word embeddings of Collobert and Weston (2008).[2]

3.2 Co-Compositionality

In order to model co-compositionality, we apply prototype projection to both the verb and the object. In particular, suppose verb is w_v and object is w_o, $\mathbf{C}_{(\textit{VerbOf},w_o)}$ is used to project w_v and $\mathbf{C}_{(\textit{ObjOf},w_v)}$ is used to project w_o. The vector that represents the overall meaning of verb-object with prototype projection is computed by:

$$\mathbf{cocomp}(w_v, w_o) = f(\mathbf{prpj}_{(\textit{VerbOf},w_o)}(w_v), \mathbf{prpj}_{(\textit{ObjOf},w_v)}(w_o)) \quad (7)$$

Function f can be a compositional computation like simple addition or element-wise multiplication of two vectors. This is graphically shown in Figure 1.

4 Unsupervised Learning of Co-Compositionality

In this section, we propose a new neural language model that learns word representations while jointly accounting for compositional semantics. One central assumption of our work (and many other works in compositional semantics) is that a single vector

[2] ronan.collobert.com/senna/

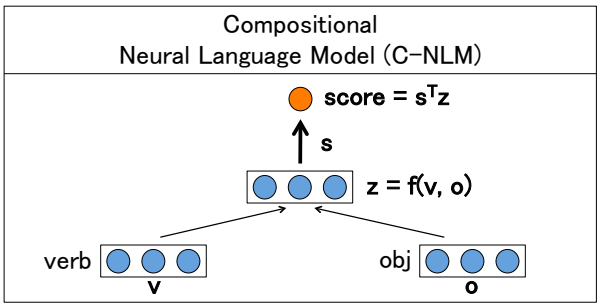

Figure 3: Compositional Neural Language Model (C-NLM).

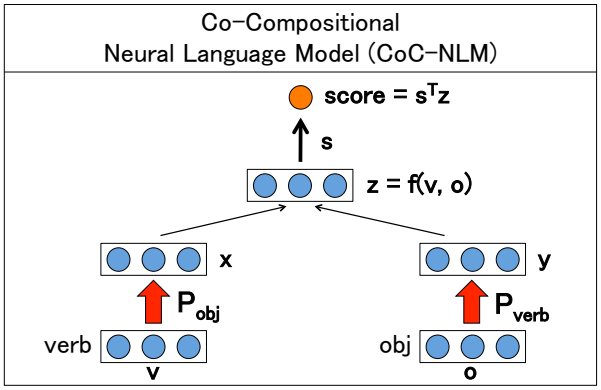

Figure 4: Co-Compositional Neural Language Model (CoC-NLM) is C-NLM with prototype projection.

per word type sufficiently represents the multiple meanings and usage patterns of a word.[3] That means that for a polysemous word, its word vector actually represents an aggregation of the distinctly different contexts it occurs in. We will show that such an assumption is quite reasonable under our model, since the prototype projections successfully tease out the proper semantics from these aggregate representations.

However, it is natural to wonder whether one can do better if one incorporates the compositional model into the training of the word representations in the first place. To do so, we formulate a novel model called *Compositional Neural Language Model* (Section 4.1). This model is a combination of an unsupervised training algorithm with basic compositionality (addition/multiplications). Then, we extend this model with the projection idea in section 3.2 to formulate a *Co-Compositional Neural Language Model* (Section 4.2).

4.1 Compositional Neural Language Model (C-NLM)

Compositional Neural Language Model (C-NLM) is a combination of a word representation learning method and compositional rule. In contrast to other compositional models based on machine learning, our model has no complex parameters for modeling composition. Composition is modeled using straightforward vector addition/multiplications; instead, what is learned is the word representation.

Figure 3 shows the C-NLM. The learning algorithm is unsupervised, and works by artificially generating negative examples in a fashion analogous to the NLM learning algorithm of (Collobert and Weston, 2008) and contrastive estimation (Smith and Eisner, 2005). First, given some initial word representations and raw sentences, we compute the compositional vector with function f (in this section, we will assume that we will be using the addition operator). Second, in order to obtain the score of compositional vector, we compute the dot product with vector $\mathbf{s} \in \mathbb{R}^n$ (n is the dimension of the word vector space): verb vector $\mathbf{v} = \mathbf{d}(w_v)$ and object vector $\mathbf{o} = \mathbf{d}(w_o)$.

$$score(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}} f(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}}(\mathbf{v} + \mathbf{o}) \qquad (8)$$

We also create a corrupted pair by substituting a random verb $w_{verb}{}'$. The cost function $J = \max(0, 1 - score(\mathbf{v}, \mathbf{o}) + score(\mathbf{v}_c, \mathbf{o}))$, where $\mathbf{v}_c$ is the word vector of $w_{verb}{}'$, encourages that the score of correct pair is higher than the score of the corrupt pair. Let $\mathbf{z} = \mathbf{v} + \mathbf{o}$, our model parameters are $\theta = (\mathbf{s}, \mathbf{z}, \mathbf{v})$. The optimization is divided into two steps:

1. Optimize $\mathbf{s}$ and $\mathbf{z}$ via SGD.

2. Let $\mathbf{z}_{new}$ be the updated $\mathbf{z}$ via step 1. The new verb vector $\mathbf{v}_{new}$ trained within additive compositionality is just $\mathbf{v}_{new} = \mathbf{z}_{new} - \mathbf{o}$. Note that if we also want to optimize $\mathbf{o}$, we may want to also corrupt the object and run SGD in step 2 as well.

4.2 Co-Compositional Neural Language Model (CoC-NLM)

We now add prototype projection into C-NLM, making our final model: *Co-Compositional Neural*

[3] There are works on *multiple* representations, e.g., (Reisinger and Mooney, 2010); we focus on single representation here.

Language Model (CoC-NLM). We define the score function as dot product of $\mathbf{s}$ and additional vector of prototype projected vectors (Figure 4). Let $\mathbf{P}_{obj} = \mathbf{P}_{(VerbOf,w_o)}$ and $\mathbf{P}_{verb} = \mathbf{P}_{(ObjOf,w_v)}$,

$$score(\mathbf{v}, \mathbf{o}) = \mathbf{s}^{\mathrm{T}}(\mathbf{P}_{obj}\mathbf{v} + \mathbf{P}_{verb}\mathbf{o}). \quad (9)$$

Let $\mathbf{x} = \mathbf{P}_{obj}\mathbf{v}$, $\mathbf{y} = \mathbf{P}_{verb}\mathbf{o}$ and $\mathbf{z} = \mathbf{x} + \mathbf{y}$. Our model parameters are $\theta = (\mathbf{s}, \mathbf{z}, \mathbf{v})$. The optimization algorithm of CoC-NLM is divided into three steps like C-NLM. First, we optimize $\mathbf{s}$ and $\mathbf{z}$. Second, the projected verb vector is updated as $\mathbf{x}_{new} = \mathbf{z}_{new} - \mathbf{y}$. Finally we optimize $\mathbf{v}$ to minimize the Euclidean distance between $\mathbf{x}_{new}$ and $\mathbf{P}_{obj}\mathbf{v}$, where λ is a regularization hyper-parameter:

$$J(\mathbf{v}) = \frac{1}{2}||\mathbf{x}_{new} - \mathbf{P}_{obj}\mathbf{v}||^2 + \frac{\lambda}{2}\mathbf{v}^{\mathrm{T}}\mathbf{v} \quad (10)$$

5 Evaluation

5.1 Dataset

In order to evaluate the performance of our new co-compositional model with prototype projection and word representation learning algorithm, we make use of the disambiguation task of transitive sentences developed by Grefenstette and Sadrzadeh (2011). This is an extension of the two words phrase similarity task defined in Mitchell and Lapata (2008), and constructed according to similar guidelines. The dataset consists of similarity judgments between a *landmark* verb and a triple consisting of a transitive target verb, subject and object extracted from the BNC corpus. Human judges give scores between 1 to 7, with higher scores implying higher semantic similarity. For example, Table 2 shows some examples from the data: we see that the verb *meet* with subject *system* and object *criterion* is judged similar to the landmark verb *satisfy* but not *visit*. The dataset contains a total of 2500 similarity judgements, provided by 25 participants.[4] The task is to have the model produce a score for each pair of landmark verb and verb-subject-object triple. Models are evaluated by computing the Spearman's ρ correlation between its similarity scores and that of the human judgments.

[4] http://www.cs.ox.ac.uk/people/edward.grefenstette/

verb	subj	obj	landmark	sim
meet	system	criterion	satisfy	6
meet	system	criterion	visit	1
write	student	name	spell	7
write	student	paper	spell	2

Table 2: Examples from the disambiguation task developed by Grefenstette and Sadrzadeh (2011). Human judges give scores between 1 to 7, with higher scores implying higher semantic similarity. Verb *meet* with subject *system* and object *criterion* is judged similar to the landmark verb *satisfy* but not *visit*.

5.2 Baselines

We compare our model against multiple baselines for semantic compositionality:

1. Mitchell and Lapata's (2008) additive and element-wise multiplicative model as simplest baselines.

2. Grefenstette and Sadrzadeh's (2011) model based on the abstract categorical framework (Coecke et al., 2010). This model computes the outer product of the subject and object vector, the outer product of the verb vector with itself, and then the element-wise product of both results.

3. Erk and Padó's (2008) model, which adapts the word vectors based on context and is the most similar in terms of motivation to ours.

4. Van de Cruy et al. (2013) multi-way interaction model based on matrix factorization. This achieves the best result for this task to date.

A detailed explanation of these models will be provided in Section 7. For the underlying word representations, we experiment with sparse 2000-dim SDS and dense 50-dim NLM. These are provided by Blacoe and Lapata (2012)[5] and trained on the British National Corpus (BNC). We are interested in knowing how sensitive each model is to the underlying word representation. In general, this is a challenging task: the upper-bound of $\rho = 0.62$ is the inter-annotator agreement.

[5] http://homepages.inf.ed.ac.uk/s1066731/index.php?page=resources

5.3 Implementation details

In terms of implementation detail, our model and our re-implementation of Erk and Pado's model make use of the ukWaC corpus (Baroni et al., 2009).[6] This corpus is a two billion word corpus automatically harvested from the web and parsed by the MaltParser (Nivre et al., 2006). We use ukWaC corpus to collect $W(\textit{VerbOf}, w_o)$ and $W(\textit{ObjOf}, w_v)$ for prototype projections. We also extract about 5000 verb-object pairs that relevant for testdata from this corpus to train our neural network learning algorithm. In our co-compositional model, the contribution ratio of SVD is set to 80% (i.e. automatically fixing k in SVD to include 80% of the top singular values). We set the number of prototype vectors to be $m = 20$, where $W(\textit{VerbOf}, w_o)$ is filtered with high frequency words and $W(\textit{ObjOf}, w_v)$ is filtered with both high frequency and high similarity words. In our model, we output the scores for SVO triple sentence dataset as (subject=w_s, verb=w_v, object=w_o, f = Addition/Multiplication):

$$\mathbf{cocomp}(w_s, w_v, w_o) = f(\mathbf{d}(w_s), \mathbf{cocomp}(w_v, w_o)) \quad (11)$$

6 Results and Discussion

6.1 Main Results: The Correlation

Table 3 shows the correlation scores of various models. Our observations are as follows:

1. The best reported result for this task (Van de Cruys et al., 2013) is $\rho = 0.37$. Our model (with NLM as word representation and f=Addition as operator) achieves $\rho = 0.44$, outperforming it by a large margin. To the best of our knowledge, this is now state-of-the-art result for this task.

2. Our model is not very sensitive to the underlying word representation. With f=Addition, we have $\rho = 0.41$ for SDS vs $\rho = 0.44$ for NLM. With f=Multiply, we have $\rho = 0.37$ for SDS vs. $\rho = 0.35$ for NLM. This implies that the prototype projection is robust to the underlying word representation, which is a desired characteristic of compositional models.

[6] http://wacky.sslmit.unibo.it/doku.php?id=corpora

Model	ρ
Grefenstette and Sadrzadeh (2011)	0.21
Add (SDS)	0.31
Add (NLM)	0.31
Multiply (SDS)	0.35
Multiply (NLM)	0.30
Van de Cruys et al. (2013)	0.37
Erk and Padó (SDS)	0.39
Erk and Padó (NLM)	0.03
Co-Comp with f=Add (SDS)	0.41
Co-Comp with f=Add (NLM)	**0.44**
Co-Comp with f=Multiply (SDS)	0.37
Co-Comp with f=Multiply (NLM)	0.35
Upper bound	0.62

Table 3: Results of the different compositionality models on the similarity task. The number of prototype words $m = 20$ in all our models. Our model (f=Addition and NLM) achieves the new state-of-the-art performance for this task ($\rho = 0.44$).

3. The contextual model of Erk and Padó (SDS) also performed relatively well ($\rho = 0.39$), in fact outperforming the Van de Cruy et al. (2013) result as well. This means that the general idea of adapting word representations based on context is a very powerful one. However, Erk and Padó's model using the NLM representation is extremely poor ($\rho = 0.03$). The reason is that it uses a product operation under-the-hood to adapt the vectors, which inherently assumes a sparse representation. In this sense, our projection approach is more robust.

The state-of-the-art result for our model in Table 3 does not yet make use of the training algorithm described in Section 4. It is simply implementing the co-compositionality idea using prototype projections (Section 3.2). Next in Section 6.2 we will show additional gains using unsupervised learning.

6.2 Improvements from unsupervised learning

In this experiment, we examine how much gain is possible by re-training the word representation of verbs using the unsupervised algorithm described in Section 4. We focus on the additive model of Compositional NLM, both basic and prototype projection. The initial word representation is from

model	original representation	re-trained
C-NLM	0.31	0.38
CoC-NLM	0.44	**0.47**

Table 4: Results of re-training the word representation for C-NLM and CoC-NLM. Learning rate $\alpha = 0.01$, regularization $\lambda = 10^{-4}$ and iteration = 20. One iteration is one run through the dataset of 5000 verb-object pairs which we made from the ukWaC corpus.

NLM. Table 4 shows the gains in correlation score.

This result shows that our learning model successfully captures good representation within co-compositionality of additive model. In contrast to other previous compositional models, our model does not require estimating a large number of parameters for computation of compositional vectors and word representation itself is more suitable for it. Furthermore, learning is very fast, taking about 10 minutes for C-NLM on a standard machine with Intel Core i7 2.93Ghz CPU and 8GB of RAM.

6.3 The number of prototype words

The number of prototype words (m in Figure 1) we use to generate the prototype space is one hyper-parameter that our model has. Here, we analyze the effect of the choice of m. Figure 5 shows the relation of m and the performance of co-compositional model with prototype projections using either SDS or NLM representations. In general, both NLM and SDS show relatively smooth and flat curves across m, indicating the relative robustness of the approach. Nevertheless, results do degrade for large m, due to increase in noise from non-prototype words. Further, it does appear that NLM has a slower drop in correlation with increasing m compared with SDS. This suggests that NLM is more robust, which is possibly attributable to the dense and low-dimensional distributed features.

6.4 Variations in model configuration

We have presented a compositional model of the form $\mathbf{d}(w_s) + \mathbf{cocomp}(w_v, w_o)$, where prototype projections are performed on both w_v and w_o and w_s is composed as is without projection. In general, we have the freedom to choose what to project and what not to project under this co-compositional framework. Here in Table 5 we show the results of

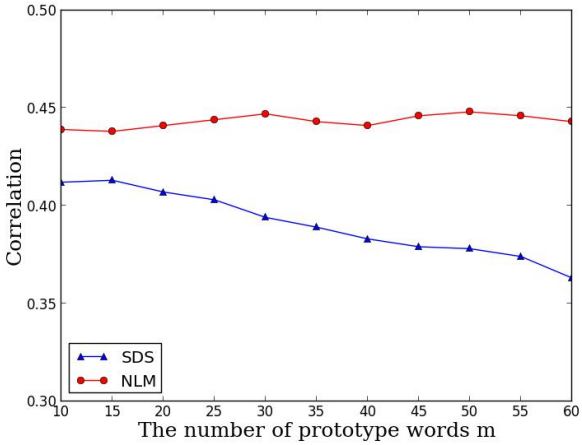

Figure 5: The relation between the number of prototype words and correlation of SDS or NLM. In general, NLM has higher correlation than SDS and is more robust across the m.

Subj	Verb	Obj	NLM ρ	SDS ρ
prpj	prpj	prpj	0.39	0.37
+	prpj	prpj	0.44	0.41
	prpj	prpj	0.45	0.41
+	prpj	+	0.43	0.38
	prpj	+	0.43	0.38
+	+	+	0.31	0.31

Table 5: Variants of the full co-compositional model, based on how subject, verb, and object vector representations are included. prpj indicates that prototype projection is used. + indicates that the vector is added without projection first. Blank indicates that the vector is not used in the final compositional score.

these variants, using f =Addition and SDS/NLM representations without re-training. We note that our positive results mainly come from the verb projections. Subject information actually does not help. We believe this best configuration is task-dependent; in this test collection, the subjects appear to have little contribution to the landmark verb.

7 Related work

In recent years, several sophisticated vector space models have been proposed for computing compositional semantics. Mitchell and Lapata (2010), Erk (2012) and Baroni et al. (2013) are recommended survey papers.

One of the first approaches is the vector addition/multiplication idea of Mitchell and Lapata (2008). The appeal of this kind of simple approach is its intuitive geometric interpretation and its robustness to various datasets. However, it may not be sufficiently expressive to represent the various factors involved in compositional semantics, such as syntax and context. To this end, Baroni and Zamparelli (2010) present a compositional model for adjectives and nouns. In their model, an adjective is a matrix operator that modifies the noun vector into an adjective-noun vector. Zanzotto et al. (2010) and Guevara (2010) also proposed linear transformation models for composition and address the issue of estimating large matrices with least squares or regression techniques. Socher et al. (2012) extend this linear transformation approach with the more powerful model of Matrix-Vector Recursive Neural Networks (MV-RNN). Each node in a parse tree is assigned both a vector and a matrix. The vector captures the actual meaning of the word itself, while the matrix is modeled as a operator that modify the meaning of neighboring words and phrases. This model captures semantic change phenomenon like *not bad* is similar to *good* due to a composition of the *bad* vector with a meaning-flipping *not* matrix. But this MV-RNN also need to optimize all matrices of words from initial value (identity plus a small amount of Gaussian noise) with supervised dataset like movie reviews. Our prototype projection model is similar to these models as a matrix-vector operation, except that the matrix is not learned and computed from prototype words. In future work, we can imagine integrating the two models, using these prototype projection matrices as initial values for MV-RNN training (Socher et al., 2012).

Another approach is exemplified by Coecke et al. (2010). In their mathematical framework unifying categorical logic and vector space models, the sentence vector is modeled as a function of the Kronecker product of its word vectors. Grefenstette and Sadrzadeh (2011) implement this based on unsupervised learning of matrices for relational words and apply them to the vectors of their arguments. Their idea is that words with relational types, such as verbs, adjectives, and adverbs are matrices that act as a filter on their arguments. They also developed a new semantic similarity task based on transitive verbs, which is the dataset we used here. The previous state-of-the-art result for this task comes from the model of Van de Cruys et al. (2013). They model compositionality as a multi-way interaction between latent factors, which are automatically constructed from corpus data via matrix factorization.

Composition Operator	Parameter
Add: $w_1 u + w_2 v$	$w_1, w_2 \in \mathbb{R}$
Multiply: $u^{w_1} \odot v^{w_2}$	$w_1, w_2 \in \mathbb{R}$
FullAdd: $W_1 u + W_2 v$	$W_1, W_2 \in \mathbb{R}^{n \times n}$
LexFunc: $A_u v$	$A_u \in \mathbb{R}^{n \times n}$
FullLex: $\sigma([W_1 A_u v, W_2 A_v u])$	$A_u, A_v \in \mathbb{R}^{n \times n}$
	$W_1, W_2 \in \mathbb{R}^{n \times n}$
Ours (Add): $P_{(R,v)} u + P_{(R,u)} v$	SVD's (m, k)
Ours (Mult): $P_{(R,v)} u \odot P_{(R,u)} v$	SVD's (m, k)

Table 6: Comparison of composition operators that combine two word vector representations, $u, v \in \mathbb{R}^n$ and their learning parameters. Our model only needs two hyper-parameters: the number of prototype words m and dimensional reduction k in SVD

Comprehensive evaluation of various existing models are reported in (Blacoe and Lapata, 2012; Dinu et al., 2013). Blacoe and Lapata (2012) highlight the importance of jointly examining word representations and compositionality operators. However, two out of three composition methods they evaluate are parameter-free, so that they can side-step the issue of parameter estimation. Dinu et al. (2013) describe the relation between word vector and compositionality in more detail with free parameters. Table 6 summarizes some ways to compose the meaning of two word vectors (u, v), following (Dinu et al., 2013). These range from simple operators (e.g. Add and Multiply) to expressive models with many free parameters (e.g. LexFunc, FullLex). Many of these models need to optimize $n \times n$ parameters, which may be large. On the other hand, our model only needs two hyper-parameters: the number of prototype words m and dimensional reduction k in SVD (Table 6). Furthermore, our model performance with neural language model word embeddings is robust to variations in m.

Most closely related to our work is the work by Erk and Padó (2008; 2009) and Thater et al. (2010; 2011), which falls under the research theme of computing *word meaning in context*. Both methods are characterized by the use of selectional prefer-

ence information for subjects, verbs, and objects in context; our prototype word vectors are essentially equivalent to this idea. The main difference is in how we modify the target word representation $\mathbf{v}$ using this information: whereas we project $\mathbf{v}$ onto a latent subspace formed by collection of prototype vectors, Erk and Padó (2008; 2009) and Thater et al. (2010; 2011) use the prototype vectors to directly modify the elements of $\mathbf{v}$, i.e. by element-wise product with the centroid prototype vector. Intuitively, both our method and theirs essentially delete part of a word vector representation to adapt the meaning in context. We believe the projection is more robust to the underlying word representation (and this is shown in the results for SDS vs. NLM representations), but we note that we may be able to borrow some of more sophisticated ways to find prototype vectors from Erk and Padó (2008; 2009) and Thater et al. (2010; 2011).

8 Conclusion and Future Work

We began this work by asking how it is possible to handle polysemy issues in compositional semantics, especially when adopting distributional semantics methods that construct only one representation per word type. After all, the different senses of the same word are all conflated into a single vector representation. We found our inspiration in Generative Lexicon Theory (Pustejovsky, 1995), where ambiguity is resolved due to *co-compositionality* of the words in the sentence, i.e., the meaning of an ambiguous verb is generated by the properties the object it takes, and vice versa. We implement this idea in a novel neural network model using *prototype projections*. The advantages of this model is that it is robust to the underlying word representation used and that it enables an effective joint learning of word representations. The model achieves the current state-of-the-art performance ($\rho = 0.47$) on the semantic similarity task of transitive verbs (Grefenstette and Sadrzadeh, 2011).

Directions for future research include:

- Experiments on other semantics tasks, such as paraphrase detection, word sense induction, and word meaning in context.
- Extension to more holistic sentence-level composition using a matrix-vector recursive framework like (Socher et al., 2012).
- Explore further the potential synergy between Distributional Semantics and the Generative Lexicon.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant Number 24800041, JSPS KAKENHI 2430057 and Microsoft Research CORE Project. We would like to thank Hiroyuki Shindo and anonymous reviewers for their helpful comments.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2013. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technologies*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *CoRR*, abs/1003.4394.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch.

Journal of Machine Learning Research, 12:2493–2537.

Georgiana Dinu, Nghia The Pham, and Marco Barori. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Katrin Erk and Sebastian Padó. 2009. Paraphrase assessment in structured vector space: Exploring parameters and datasets. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

G Frege. 1892. Über sinn und bedeutung. In *Zeitschfrift für Philosophie und philosophische Kritik, 100*.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the Workshop on GEometrical Models of Natural Language Semantics*.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

James Pustejovsky. 1995. *The Generative Lexicon*. MIT Press, Cambridge, MA.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Asian Federation of Natural Language Processing (IJCNLP)*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37(1):141–188.

Tim Van de Cruys, Thierry Poibeau, and Anna Korhonen. 2013. A tensor-based factorization model of semantic compositionality. In *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.