Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Non-Linear Similarity Learning for Compositionality

Masashi Tsubaki, Kevin Duh*, Masashi Shimbo, Yuji Matsumoto
Nara Institute of Science and Technology, *Johns Hopkins University
{masashi-t,shimbo,matsu}@is.naist.jp, *kevinduh@cs.jhu.edu

Abstract

Many NLP applications rely on the existence of similarity measures over text data. Although word vector space models provide good similarity measures between words, phrasal and sentential similarities derived from composition of individual words remain as a difficult problem. In this paper, we propose a new method of of non-linear similarity learning for semantic compositionality. In this method, word representations are learned through the similarity learning of sentences in a high-dimensional space with kernel functions. On the task of predicting the semantic relatedness of two sentences (SemEval 2014, Task 1), our method outperforms linear baselines, feature engineering approaches, recursive neural networks, and achieve competitive results with long short-term memory models.

1 Introduction

The notion of semantic similarity between text data (e.g., words, phrases, sentences, and documents) plays an important role in natural language processing (NLP) applications such as information retrieval, classification, and extraction. The simplest similarity measurement is based on word matchings rather than word meanings, and suffers from lack of generalization.

Recently, word semantic vector spaces based on distributional and distributed models have become popular (Turney 2012; Collobert et al. 2011; Pennington, Socher, and Manning 2014). While such word representations are sufficient to compute the similarity between words, it is not trivial to capture meaning of phrases and sentences composed of individual words. To overcome the weakness, modeling and learning compositionality have received a lot of attention (Mitchell and Lapata 2010). The goal of this research is to formulate how word vectors and operations are learned and modeled to properly represent phrasal and sentential semantics. However, there are still some problems in compositional vector semantics. In particular, we are concerned with the following question:

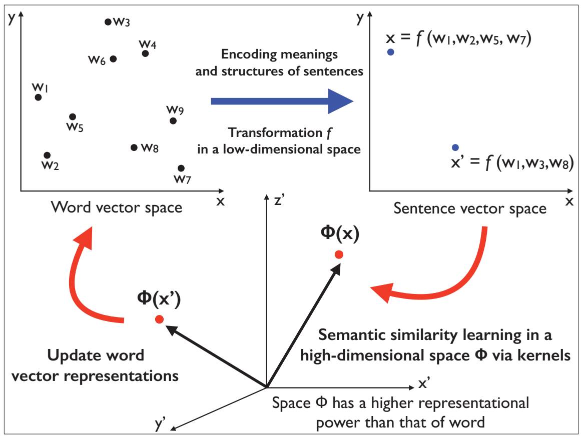
y
w3
w4
w6
w1
w5
w9
w8
w2
w7
x
Word vector space
Encoding meanings and structures of sentences
Transformation f in a low-dimensional space
y
x = f (w1,w2,w5, w7)
x' = f (w1,w3,w8)
x
Sentence vector space
z'
Φ(x)
Φ(x')
Update word vector representations
Semantic similarity learning in a high-dimensional space Φ via kernels
x'
y'
Space Φ has a higher representational power than that of word

Figure 1: Overview of our method. Our aim is to design a word representation learning architecture which combines the simplicity of encoding sentence structures in a low-dimensional space with the power derived from a non-linear similarity learning for the sentence semantics in a high-dimensional space via kernel functions.

> How can we represent the meaning of sentences, which cover richer meaning variations than that of words, in a vector space?

To answer this question, we propose a new method of non-linear similarity learning for semantic compositionality. Our approach is to capture semantics of sentences in a space different from that of words. Presumably, this space must be of higher dimension than the word space, since a sentence contains far more information than words. For example, a sentence *"Newton was inspired to formulate gravitation by watching the fall of an apple from a tree."* should have a more complex (therefore, higher-dimensional) representation than words *"apple"*, *"gravitation"*, *"formulate"*, and *"by"*.

The proposed method is inspired by the previous work on distance metric and similarity learning (Bellet, Habrard, and Sebban 2013), and in particular leverages the non-linear capacity of kernels (Kedem et al. 2012). Our goal is to obtain new word representations that account for compositionality, through the similarity learning of sentential meaning in the kernel-induced high-dimensional space. Figure 1 shows the geometric intuition of our model architecture. We can learn and obtain new word representations inexpensively by using kernel functions without explicit computation of sentence vectors in the high-dimensional space. Note that our approach differs from that of deep learning, such as recursive neural network (RNN) (Socher et al. 2014) and long short-term memory (LSTM) (Tai, Socher, and Manning 2015). RNN and LSTM use intricate neural network architectures to model and learn semantics of sentences in a low-dimensional space, whereas ours integrates both low- and high-dimensional space.

Our contributions are two-fold:

1. To the best of our knowledge, ours is the first work that addresses compositionality by combining similarity learning in kernel-induced high-dimensional space, and structure embedding in a low-dimensional space.
2. Our method is simple and effective. It outperforms linear baselines, feature engineering approaches, RNNs, and achieves competitive results with LSTM models on the task of predicting the semantic relatedness of two sentences.

2 Background

2.1 Compositional Vector Semantics

Word vector representations can be learned through many different approaches, such as distributional models using co-occurrence statistics of a word and its context, and distributed models using neural networks (Turney 2012; Collobert et al. 2011). Despite their usefulness, these word representations do not capture semantic compositionality. As a result, modeling and learning compositional semantics in the word vector space have emerged as another important line of research (Mitchell and Lapata 2010). For phrases and sentences, many different models have been explored (Socher et al. 2013; Tsubaki et al. 2013).

In particular, recursive neural networks (RNNs) are used to represent a sentence vector. RNNs use the following composition function to compute a phrase vector $\mathbf{p}$ from two d dimensional vectors $\mathbf{d}(w_i)$ and $\mathbf{d}(w_j)$ of words w_i and w_j:

$$\mathbf{p} = g\left(\mathbf{W}\begin{bmatrix}\mathbf{d}(w_i)\\ \mathbf{d}(w_j)\end{bmatrix}\right), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d\times 2d}$ is the weight parameter to learn, and g is a non-linear function such as sigmoid or $\tanh$. To obtain a vector representation of a sentence, we run a parser on the sentence, and apply RNNs recursively at each node of the parse tree, using phrase vectors in place of $\mathbf{d}(w_i)$, or $\mathbf{d}(w_j)$ whenever necessary.

2.2 Distance Metric and Similarity Learning

The notion of the metric plays an important role in machine learning (Bellet, Habrard, and Sebban 2013). There exists a line of research on learning a measure of distance or similarity between data that is suitable for problems at hand.

For example, the distance metric learning (Xing et al. 2002) is to optimize the Mahalanobis distance: $D_{\mathbf{M}}(\mathbf{x},\mathbf{x}') = (\mathbf{x}-\mathbf{x}')^{\mathrm{T}}\mathbf{M}(\mathbf{x}-\mathbf{x}')$, and the similarity learning (Qamar et al. 2008) is to optimize the inner product: $K_M(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{M}\mathbf{x}'$, where $\mathbf{x}$ and $\mathbf{x}'$ are feature vectors, and $\mathbf{M}$ is a transformation matrix to learn. Here, by decomposing positive semi-definite matrix $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W}$, we can reformulate the above equations as follows:

$$D_{\mathbf{M}}(\mathbf{x},\mathbf{x}') = ||\mathbf{W}\mathbf{x} - \mathbf{W}\mathbf{x}'||^2, \tag{2}$$

$$K_{\mathbf{M}}(\mathbf{x},\mathbf{x}') = (\mathbf{W}\mathbf{x})^{\mathrm{T}}(\mathbf{W}\mathbf{x}'). \tag{3}$$

From this perspective, distance metric and similarity learning are equivalent to learning the linear projection $\mathbf{W}$, which maps $\mathbf{x}$ and $\mathbf{x}'$ into new representations. Furthermore, these learning techniques can be extended with kernel methods (Kedem et al. 2012). The theory describes that the kernel function K implicitly maps original input data set $\mathcal{X}$ to a high-dimensional (possibly infinite) reproducing kernel Hilbert space (RHKS) $\mathcal{H}$ through $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{H}$ and computes the inner product therein; i.e., $K(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}'),\ \ \mathbf{x},\mathbf{x}' \in \mathcal{X}$. The motivation of kernelization is to obtain a good high-dimensional space for solving problems while maintaining computational tractability.

2.3 Neural Networks for Paired Data

Recently, there has been growing interest in applying neural networks to various paired data. For example, (Gao et al. 2014) construct a neural network for machine translation with parallel training data. This model represents a pair of source and target phrases in a common low-dimensional space, and their translation score is computed and optimized using the cosine similarity between the pair of vectors. (Kiela and Bottou 2014) construct a neural network for multi-modal representations by integrating linguistic vectors with visual concept vectors. These methods allow us to learn richer meaning representations by optimizing the Euclidean distance and cosine similarity for paired data.

Note that, from the distance metric and similarity learning perspective, these methods are equivalent to learning a non-linear transformation function $||N(\mathbf{x}) - N(\mathbf{x}')||^2$ and $N(\mathbf{x})^{\mathrm{T}}N(\mathbf{x}')$, where N is a non-linear function, and $\mathbf{x}$ and $\mathbf{x}'$ are feature vectors of paired data. Typically, these functions are optimized with neural networks (Wu et al. 2013).

3 Method

In this section, we introduce a method of non-linear similarity learning for semantic compositionality, and discuss the motivation behind it. Consider a training dataset that contains N pairs of tuples $\{((S_i, S'_i), y_i)\}_{i=1}^{N}$, where (S_i, S'_i) is a sentence pair, and $y_i \in [0.0, 1.0]$ is the normalized similarity score of (S_i, S'_i). The goal of this task is to correctly predict similarity scores for new sentence pairs. Our aim is to design a model which jointly learns low-dimensional word representation (Section 3.1) and the semantic similarity of sentences in high-dimensional space (Section 3.2). Our approach is motivated by the following hypothesis:

For similarity computations that require composition (e.g., sentence similarity), it is necessary to map the compositional representation to a space with higher representational power than the original low-dimensional space of the components (e.g., words).

Under this hypothesis, the goal of compositional semantics is to model how high-dimensional sentence similarity is computed from low-dimensional word representations. This goal differs from that of the current neural network-based models, which attempt to embed and measure the similarity of sentences in the same low-dimensional space of words. Our motivation of kernelization in compositionality is to obtain a good high-dimensional space for representing richer meanings of sentences while maintaining computational tractability.

3.1 Encoding of Sentence Meaning in Low-Dimensional Space

We compute a vector representation $\mathbf{x}$ of a sentence S using a function f for encoding its meaning; i.e., $\mathbf{x} = f(S)$. In this paper, we exploit two functions. The simplest candidate of f is: $f_{SUM}(S) = \sum_{w \in S} \mathbf{d}(w)$, where w is a word in the sentence S, and $\mathbf{d}(w) \in \mathbb{R}^d$ is a d-dimensional vector of the word w. This is a bag-of-words approach in which the sentence structure is not taken into account. For another candidate of f, we make use of syntactic and semantic structures of S such as part-of-speech (POS) tags and predicate-argument structures, which are available from the output of a POS tagger or a syntactic/semantic parser. This function is intended to capture structured sentence semantics from the concatenated word vectors. This alternative f has several variations. The first one uses POS tags, and is defined as follows:

$$f_{STR}^{POS2}(S) = g\left(\sum_{(w_i,w_j)\in P(S)} h\left(\mathbf{W}_{(pos(w_i),pos(w_j))}\begin{bmatrix}\mathbf{d}(w_i)\\ \mathbf{d}(w_j)\end{bmatrix}\right)\right),$$

where $P(S)$ is the set of word pairs in S that are related syntactically or semantically as deemed by the parser, w_i is the i-th word in the sentence S, g and h are functions of identity (i.e., linear) or elementwise $\tanh$ which is widely-used in neural networks (i.e., non-linear), and $\mathbf{W}_{(pos(w_i),pos(w_j))} \in \mathbb{R}^{d\times 2d}$ is the weight matrix to learn which depends on pairs of POS tags for corresponding words. The second variation, $f_{STR}^{SEM2}(S)$, is the same as the f_{STR}^{POS2} except that the matrices $\mathbf{W}_{(pos(w_i),pos(w_j))}$ are replaced with matrices based on the semantic relation labels from predicate-argument structures (e.g., logical subject and object) of words. Furthermore, if $P(S)$ has the set of triples (subject, verb, object) in S, we can define the third variation $f_{STR}^{SEM3}(S)$ as,

$$f_{STR}^{SEM3}(S) = g\left(\sum_{(w_i,w_j,w_k)\in P(S)} h\left(\mathbf{W}'\begin{bmatrix}\mathbf{d}(w_i)\\ \mathbf{d}(w_j)\\ \mathbf{d}(w_k)\end{bmatrix}\right)\right),$$

where $\mathbf{W}' = \mathbf{W}_{(sem(w_i),sem(w_j)),sem(w_k))} \in \mathbb{R}^{d\times 3d}$. This function naturally focuses on the action and agents in a sentence, and allow us to directly capture the meaning such as subject-verb-object triplet *"man-performing-wheelie"* in *"A man in a blue jumpsuit is courageously performing a wheelie on a motorcycle"*, and *"girl-raising-arm"* in *"A girl in a uniform, which is blue, is quickly raising her arm"*. Similarly, we can think of f_{STR}^{POS3}, which is identical to f_{STR}^{POS2} but uses triples. We finally compute the $f_{STR}^{POS}(S) = f_{STR}^{POS2}(S) + f_{STR}^{POS3}(S)$ and $f_{STR}^{SEM}(S) = f_{STR}^{SEM2}(S) + f_{STR}^{SEM3}(S)$.

While the choice of composition functions g and h further gives rise to numerous variants such as tensors and deep non-linear functions (Socher et al. 2013), we leave these to future work.

3.2 Non-linear Similarity Learning in High-Dimensional Space

In this section, we present a non-linear similarity learning method with kernel functions. We use several kernel functions. The normalized inner product, i.e., cosine similarity, is employed as the most basic kernel K between sentence vectors $\mathbf{x}$ and $\mathbf{x}'$ (described in Section 3.1): $K_{cos}(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{x}'/||\mathbf{x}||||\mathbf{x}'||$. Note that all kernel functions K, including the non-linear ones we introduce below, are normalized for simplicity and preventing the kernel value from growing out of control during learning process. When K is a non-linear kernel, the normalized kernel is the cosine similarity in the induced kernel space:

$$K_{cos}(\boldsymbol{\phi}(\mathbf{x}),\boldsymbol{\phi}(\mathbf{x}')) = \frac{K(\mathbf{x},\mathbf{x}')}{\sqrt{K(\mathbf{x},\mathbf{x})}\sqrt{K(\mathbf{x}',\mathbf{x}')}}, \quad (4)$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{\phi}$ is a feature vector of $\mathbf{x}$ in the kernel space.

In addition, two non-linear kernels are employed in this paper, polynomial kernel K_{poly} and Gaussian kernel K_{rbf} [1], defined as follows:

$$K_{poly}(\mathbf{x},\mathbf{x}') = \left(\frac{c + K_{cos}(\mathbf{x},\mathbf{x}')}{c+1}\right)^p, \quad s.t \quad c \geq 0,\ \ p \in \mathbb{N}. \quad (5)$$

$$K_{rbf}(\mathbf{x},\mathbf{x}') = \exp\left(\frac{K_{cos}(\mathbf{x},\mathbf{x}') - 1}{\sigma^2}\right), \quad s.t \quad \sigma \geq 0. \quad (6)$$

As mentioned earlier, we use the normalized version of K_{poly} and K_{rbf}, given by substituting them for K in equation (4).

By combining these kernels with the composition functions f_{SUM} and f_{STR} introduced in Section 3.1, we define the similarity of two sentences S and S' as follows:

$$Sim_{SUM}(S,S') = K(f_{SUM}(S), f_{SUM}(S')), \quad (7)$$

$$Sim_{SUM+STR}(S,S') = K(f_{SUM}(S), f_{SUM}(S')) + K(f_{STR}(S), f_{STR}(S')). \quad (8)$$

[1]Note that Gaussian kernel based on Euclidean distance is $K_{rbf}(\mathbf{x},\mathbf{x}') = \exp\left(-||\mathbf{x}-\mathbf{x}'||^2/2\sigma^2\right)$. Using $||\mathbf{x}-\mathbf{x}'||^2 = \mathbf{x}^{\mathrm{T}}\mathbf{x} + \mathbf{x}'^{\mathrm{T}}\mathbf{x}' - 2\mathbf{x}^{\mathrm{T}}\mathbf{x}'$ and substituting $K_{cos}(\mathbf{x},\mathbf{x}')$ in place of the inner product, we obtain $||\mathbf{x}-\mathbf{x}'||^2 = 2 - 2K_{cos}(\mathbf{x},\mathbf{x}')$. Thus, we use equation (8) as the Gaussian kernel in this paper.

Now, the training objective is to minimize $L(\Theta)$ for training dataset $\{((S_i, S'_i), y_i)\}_{i=1}^N$ as follow:

$$L(\Theta) = \sum_{i=1}^{N} \frac{1}{2}\{y_i - \mathit{Sim}(S_i, S'_i)\}^2 + \frac{\lambda}{2}||\Theta||^2, \tag{9}$$

where Θ is the set of all parameters: all word vector representations $\mathbf{d}(w)$ (these are the main learning parameters in the model), the weight matrix $\mathbf{W}$ (described in Section 3.1), and parameters in kernel functions (c in polynomial and σ in Gaussian). Thus, we can obtain the non-linear similarity between two sentences, and the computation and learning are done inexpensively through a kernel function in the high-dimensional space. Figure 1 shows the geometric intuition of the model architecture.

As the main part of the learning process, we compute the derivative $\partial L/\partial \mathbf{d}(w)$, optimize the loss function with the gradient descent, and obtain new word vector representations $\mathbf{d}(w)_{new}$. The gradient of polynomial and Gaussian kernels are as follows:

$$\frac{\partial K_{\mathit{poly}}}{\partial \mathbf{d}(w)} = p\left(\frac{c + K_{\mathit{cos}}}{c+1}\right)^{(p-1)}\left(\frac{1}{c+1}\right)\frac{\partial K_{\mathit{cos}}}{\partial \mathbf{d}(w)},$$

$$\frac{\partial K_{\mathit{rbf}}}{\partial \mathbf{d}(w)} = \exp\left(\frac{K_{\mathit{cos}} - 1}{\sigma^2}\right)\left(\frac{1}{\sigma^2}\right)\frac{\partial K_{\mathit{cos}}}{\partial \mathbf{d}(w)},$$

where $\partial K_{\mathit{cos}}/\partial \mathbf{d}(w)$ is the gradient of the cosine by the word vector [2]. Also, we compute a derivative for the weight matrix $\mathbf{W}$:

$$\frac{\partial K}{\partial \mathbf{W}} = K' \frac{\partial K_{\mathit{cos}}}{\partial \mathbf{W}},$$

where K' is the gradient of the non-linear kernel functions of polynomial and Gaussian, and $\partial K_{\mathit{cos}}/\partial \mathbf{W}$ is the gradient of the cosine by the weight matrix. Note that, from the metric learning perspective, optimization of this weight matrix is equivalent to learning the transformation matrix for original feature vectors (see Section 2.2)

4 Related Work

Successful approaches for the SemEval 2014 semantic relatedness subtask (describe in next section) combine several kinds of features frequently used in NLP tasks, such as surface form overlap, lexical distance, and WordNet. Some of the submitted systems show that purely compositional models reach performance above Pearson correlation r about 0.70, and these scores are lower than the one of the best purely non-compositional system which reaches r over 0.80 (Marelli et al. 2014). Note that all top systems (Zhao, Zhu, and Lan 2014; Bjerva et al. 2014; Jimenez et al. 2014; Lai and Hockenmaier 2014) are heavily feature engineered, and use external resources.

The most common way to build a sentence representation from words is to simply average the word vectors. While this bag-of-words (BOWs) model can yield reasonable performance in some tasks, it cannot distinguish sequences and structures of the sentence. Unlike the BOWs model, Recursive Neural Networks (RNNs) (Socher et al. 2011) combine word vectors in constituency trees of sentences which have potentially many hidden layers. While the induced sentence vectors work very well on many tasks, they must also take into account syntactic structure of the sentence in detail (Socher et al. 2012). Unlike previous RNNs which use constituency trees, Dependency Tree-RNN (DT-RNN) models (Socher et al. 2014) naturally focus on the action and agents in a sentence. They are better able to abstract from the details of word order and syntactic expression using dependency relations. On the other hand, in Long Short-Term Memory (LSTM) models (Hochreiter and Schmidhuber 1997), while the standard composes its hidden state from the input at the current time step and the hidden state of the unit in the previous time step, the Constituency and Dependency Tree-LSTM (Tai, Socher, and Manning 2015) composes its state from an input vector and the hidden states of arbitrarily many child units. These Tree-LSTM models builds on the RNNs, which subsequently abbreviate as Tree-RNNs in order to avoid confusion with Recurrent Neural Networks.

These deep learning and our approach have several important similarities and differences in terms of non-linearity and high-dimensionality for modeling and learning compositional semantics in a vector space. RNN and LSTM models use a composition function and apply these recursively inside a parse tree to compute a sentence vector. Note that this leads us to represent sentence vector in a low-dimensional space (e.g., the same dimensionality as words). In contrast, our method is not subjected to dimensional restraints which come from modeling with neural networks, and can flexibly represent and properly learn the sentence meaning with kernels.

5 Experiment

Dataset, Implementation, Compared Models

We evaluate our method in the SemEval 2014 semantic relatedness task, which uses the Sentences Involving Compositional Knowledge (SICK) dataset[3] (Marelli et al. 2014). The task is to predict the relatedness of two sentences, as judged by human annotators on a continuous scale from 1.0 (indicating that the two sentences are completely dissimilar) to 5.0 (indicating that the two sentences are very similar). The dataset consists of 9927 sentence pairs in a 4500/500/4927 train/dev/test split. Models are evaluated by computing Pearson's r, and Spearman's ρ correlations, and Mean Squared Error (MSE) between the gold similarity scores and scores predicted with the models.

For the proposed models, the final training objective is to minimize the loss function $L(\Theta)$ in equation (9). Θ is the set of learning parameters in our model: the word vector representation $\mathbf{d}(w)$ which is the main learning pa-

[2]Note that, for the concatenated vector $\begin{bmatrix}\mathbf{d}(w_i)\\ \mathbf{d}(w_j)\end{bmatrix}$, we can also compute the derivative, learn the phrase representations, and update word representations. But in our experiment using compositional function f_{STR}, we only learn the weight matrix $\mathbf{W}$ to prevent overfitting (Socher et al. 2014; Tai, Socher, and Manning 2015).

[3]`http://alt.qcri.org/semeval2014/task1/`

Method	r	ρ	MSE
Cosine (SUM)	0.7588	0.7391	0.4820
Poly (SUM)	0.8332	0.7810	0.3205
RBF (SUM)	0.8339	0.7804	0.3162
Cosine (SUM + STR_POS)	0.7510	0.7429	0.4510
Poly (SUM + STR_POS)	0.8301	0.7858	0.3176
RBF (SUM + STR_POS)	0.8325	0.7721	0.3094
Cosine (SUM + STR_SEM)	0.7647	0.7435	0.4787
Poly (SUM + STR_SEM)	**0.8480**	**0.7968**	**0.2904**
RBF (SUM + STR_SEM)	0.8447	0.7923	0.2968

Table 1: Correlations of with gold standard and MSE. Compositional function g and h are identity (i.e., linear). Polynomial kernel achieves the best result. The word representation is GloVe.

rameter, the weight matrix $\mathbf{W}$, and parameters in polynomial and Gaussian kernels (i.e., c and σ). We used 50-dimensional word representation, with several different initializations: random initialization within $(-0.1, 0.1)$, LSA (Turney 2012), NLM[4], and GloVe[5]. To obtain the predicate argument structure of sentence for computing f_{STR}, we used Enju[6]. We initialized all weight matrices as $\mathbf{W} = \mathbf{I} + \epsilon$, where ϵ is a matrix of small Gaussian noise, and parameters in kernels as $c = 1.0$ in polynomial and $\sigma = 1.0$ in RBF. Our models were trained using the adaptive gradient method AdaGrad (Duchi, Hazan, and Singer 2011) with learning rates $\alpha = 0.5$ for the word representations, $\beta = 10^{-2}$ for the weight matrix, and $\gamma = 10^{-3}$ for parameters in kernels, with the regularization parameter $\lambda = 10^{-6}$. These hyper-parameters $(\alpha, \beta, \gamma, \lambda)$ for our models were tuned on the development set.

We compare our method against the top systems for the SemEval 2014 semantic relatedness subtask: ECNU, The Meaning Factory, UNAL-NLP, and Illinois-LH (Zhao, Zhu, and Lan 2014; Bjerva et al. 2014; Jimenez et al. 2014; Lai and Hockenmaier 2014), which used a various types of handicrafted features. We also compare with models based on recursive neural networks (RNNs) and long short-term memory (LSTM) (Socher et al. 2014; Tai, Socher, and Manning 2015), which use deep learning architectures for computing sentence semantic representations in a low-dimensional space.

6 Results and Discussion

6.1 Main Results: Linear vs. Non-linear Similarity Measures

Table 1 shows r, ρ, and MSE for different composition and kernel models. We found that both Pearson's r and Spearman's ρ are higher with two non-linear (polynomial and Gaussian) kernels than the linear (cosine) kernel by a large margin. In particular, the model with polynomial kernel of degree $p = 4$ and SUM + STR_SEM achieved the best result among our models, Pearson's r is 0.8480, Spearman's ρ is 0.7968, and MSE is 0.2904. These results suggest mean that the similarity learning in kernel-induced high-dimensional space is effective, and the kernel space allows us to obtain new word representations which are suitable for sentence composition.

[4]http://ronan.collobert.com/senna/

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

[6]http://www.nactem.ac.uk/enju/index.html

	r (g and h are identity)	r (g and h are $\tanh$)
Cosine	0.7647	0.7717
Poly	0.8480	0.8392
RBF	0.8447	0.8393

Table 2: The non-linearity in similarity (i.e., polynomial and RBF) is effective, but non-linearity in composition (i.e., $\tanh$) does not improve results with non-linear kernels.

While the STR model (i.e., taking into account sentence structures) achieved higher performance than the SUM model (i.e., bag of words), the results of SUM with non-linear kernels are also high (polynomial: 0.8332; Gaussian: 0.8339), even though the model ignores the sentence structures. This suggests that the high-dimensional kernel space is sufficient to capture sentence meanings without modeling the structures in details. In addition, the result implies that the co-occurrence information of words in a sentence is important to capture the meaning.

6.2 Linear or Non-linear Composition

Table 2 shows the Pearson correlations for combinations of the linear or non-linear composition and similarity (all models are SUM + STR_SEM). These results imply that the non-linearity with kernel mappings is more effective than the non-linearity with $\tanh$ functions, and the combination of linear composition and non-linear similarity performs best. This combination is easy to implement and optimize because we can model and learn new word representations without constructing intricate neural networks in a low-dimensional space.

6.3 Comparison to Competitive Performers

Table 3 shows the comparison to competitive performers. Firstly, our models outperform four of the top systems submitted to SICK in SemEval 2014. Note that these four are heavily feature engineered systems, whereas our approach is mainly dependent on learning of word representations, and does not require a large number of features and external resources. Secondly, our correlation scores are higher than RNN models, DT-RNN and SDT-RNN (Socher et al. 2014). In fact, our model using the non-linear composition and linear similarity is similar to the RNN models (Socher et al. 2014), and the results are also similar to ours. This shows that our model is more robust than RNNs. Thirdly, in terms of the best correlation score, our model is the competitive with LSTM models. These LSTM models also compute a sentence representation in a low-dimensional space in the RNNs fashion. This result also shows the effectiveness of ours.

Method	r	ρ	MSE
Illinois-LH_run1 (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP_run1 (Jimenez et al. 2014)	0.8043	0.7458	0.3593
Meaning_Factory_run1 (Bjerva et al. 2014)	0.8268	0.7722	0.3224
ECNU_run1 (Zhao, Zhu, and Lan 2014)	0.8280	0.7689	0.3250
DT-RNN (Socher et al. 2014)	0.7923	0.7319	0.3822
SDT-RNN (Socher et al. 2014)	0.7900	0.7304	0.3848
LSTM	0.8528	0.7911	0.2831
Bidirectional LSTM (Graves, Jaitly, and Mohamed 2013)	**0.8567** (2)	**0.7966** (3)	**0.2736** (2)
2-layer LSTM (Graves, Jaitly, and Mohamed 2013)	0.8515	0.7896	0.2838
2-layer Bidirectional LSTM (Graves, Jaitly, and Mohamed 2013)	0.8558	0.7965	0.2762
Constituency Tree LSTM (Tai, Socher, and Manning 2015)	**0.8582** (3)	**0.7966** (3)	**0.2734** (3)
Dependency Tree LSTM (Tai, Socher, and Manning 2015)	**0.8676** (1)	**0.8083** (1)	**0.2532** (1)
Our best model	0.8480	**0.7968** (2)	0.2904

Table 3: Comparison to competitive performers. Results are grouped as follows: SemEval 2014 submissions, RNN models, Sequential and structural LSTM and Our best model. Our best result outperforms the four of the top systems submitted to SemEval and RNNs, and close to results of LSTM moddels. (1), (2), and (3) in the table are the ranking.

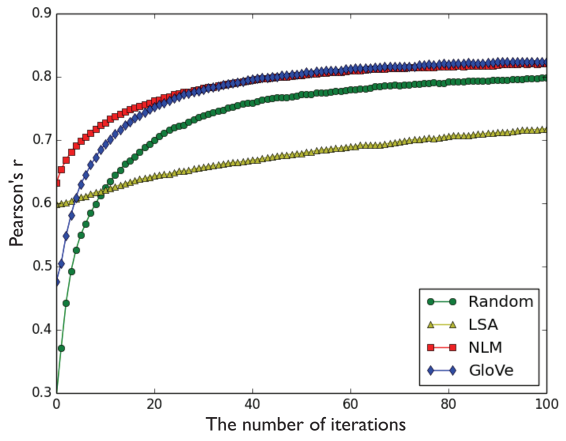

Figure 2: Learning curve with various word representations. The composition is SUM, and the kernel is polynomial.

6.4 Initialization of Word Representations

We further analyze the influence of using pre-trained word representations for initialization. Figure 2 shows that NLM and GloVe achieved good results. In addition, we also obtain, after a sufficient number of iterations, high performance with random initialization. This suggests that the representation learning with kernels is still able to achieve comparable accuracy without the help of pre-training, and we need to re-train representations for solving problems.

On the one hand, the Pearson's r with LSA is lower than others. Interestingly, although initial correlation with LSA is higher than other representations and similar to that of NLM, increasing the number of iterations does not improve the result. While initial Pearson's r of random and GloVe are lower than that of LSA and NLM, these finally achieve high Pearson's r as learning proceeds. This suggests that the LSA representation based on co-occurrence counts is not suitable for re-training with such a representation learning model. Distributed representations such as NLM and GloVe are seem more suitable.

7 Conclusion and Future Work

In this paper, we have proposed a new method of non-linear similarity learning for semantic compositionality. Instead of relying on RNNs operating in a low-dimensional space, we train kernel functions that allows us to measure the semantic similarity of sentences in a high-dimensional space. In the task of predicting the relatedness of two sentences, our method outperformed linear baselines, feature engineering approaches, RNNs, and achieved competitive results with LSTMs. It is conceivable that combining our similarity learning approach with LSTMs could lead to even more expressive models of compositionality, transitioning between higher-order sentence representations and lower-order word vector spaces. This is the main interest of our future work.

References

Bellet, A.; Habrard, A.; and Sebban, M. 2013. A survey on metric learning for feature vectors and structured data. *CoRR* abs/1306.6709.

Bjerva, J.; Bos, J.; van der Goot, R.; and Nissim, M. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *SemEval*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research (JMLR)*.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Gao, J.; He, X.; Yih, W.-t.; and Deng, L. 2014. Learning continuous phrase representations for translation modeling.

In *Proceedings of the Conference on Association for Computational Linguistics (ACL)*.

Graves, A.; Jaitly, N.; and Mohamed, A.-R. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU)*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jimenez, S.; Duenas, G.; Baquero, J.; Gelbukh, A.; Bátiz, A. J. D.; and Mendizábal, A. 2014. Unal-nlp: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *SemEval*.

Kedem, D.; Tyree, S.; Sha, F.; Lanckriet, G. R.; and Weinberger, K. Q. 2012. Non-linear metric learning. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

Kiela, D., and Bottou, L. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Lai, A., and Hockenmaier, J. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *SemEval*.

Marelli, M.; Bentivogli, L.; Baroni, M.; Bernardi, R.; Menini, S.; and Zamparelli, R. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *SemEval*.

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

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Qamar, A. M.; Gaussier, E.; Chevallet, J.-P.; and Lim, J. H. 2008. Similarity learning for nearest neighbor classification. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*.

Socher, R.; Lin, C. C.; Manning, C.; and Ng, A. Y. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Socher, R.; Huval, B.; Manning, C. D.; and Ng, A. Y. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Socher, R.; Le, Q. V.; Manning, C. D.; and Ng, A. Y. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics (TACL)*.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tsubaki, M.; Duh, K.; Shimbo, M.; and Matsumoto, Y. 2013. Modeling and learning semantic co-compositionality through prototype projections and neural networks. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Turney, P. D. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research (JAIR)*.

Wu, P.; Hoi, S. C.; Xia, H.; Zhao, P.; Wang, D.; and Miao, C. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the ACM international conference on Multimedia*.

Xing, E. P.; Jordan, M. I.; Russell, S.; and Ng, A. Y. 2002. Distance metric learning with application to clustering with side-information. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

Zhao, J.; Zhu, T. T.; and Lan, M. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *SemEval*.