Adaptive Multi-Compositionality for Recursive Neural Network Models

Li Dong, Furu Wei, Ke Xu, Shixia Liu, and Ming Zhou

***Abstract*—Recursive neural network models have achieved promising results in many natural language processing tasks. The main difference among these models lies in the composition function, i.e., how to obtain the vector representation for a phrase or sentence using the representations of words it contains. This paper introduces a novel Adaptive Multi-Compositionality (AdaMC) layer to recursive neural network models. The basic idea is to use more than one composition function and adaptively select them depending on input vectors. We develop a general framework to model the semantic composition as a distribution of these composition functions. The composition functions and parameters used for adaptive selection are jointly learnt from the supervision of specific tasks. We integrate AdaMC into existing recursive neural network models and conduct extensive experiments on the Stanford Sentiment Treebank and semantic relation classification task. The experimental results demonstrate that AdaMC improves the performance of recursive neural network models and outperforms the baseline methods.**

***Index Terms*—Compositionality, neural network, recursive neural network (RNN), semantic composition.**

I. Introduction

RECURSIVE neural network models (RNNMs), which utilize recursive structures of inputs (e.g., sentences) to obtain their semantic representations, are one family of popular deep learning models. They are particularly effective for many natural language processing tasks due to the compositional nature of natural language. The basic idea of RNNMs is to represent text at different granularity (e.g., words, phrases, and sentences) as low-dimensional dense vectors. Using word vectors as basic units, the meaning representation of a longer text can be recursively inferred by **semantic composition** of the vectors of words or phrases it contains [1]. The composition algorithm is the key to make vector representations go beyond words to phrases and sentences. As illustrated in Fig. 1, RNNMs compute the vector representation of phrase "*not so good*" via combining the vectors of "*not*" and "*so good*", and the vector "*so good*" is obtained by word vectors of "*so*" and "*good*". In a similar way, vector representations of sentences can be computed. The main difference among RNNMs lies in semantic composition methods, namely, how to obtain the vector representation for a phrase or sentence using the representations of words and phrases it contains.

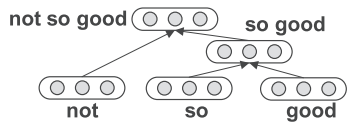

Fig. 1. Vector representations of phrases are computed recursively in Recursive Neural Network Models.

Recently, there have been several attempts in literature to address semantic composition for RNNMs. For example, RNN [2] uses a global matrix to linearly combine elements of vectors, while RNTN [3] employs a global tensor to model interactions of dimensions. Intuitively, we can employ multiple composition functions, instead of only using a single global one. As shown in Fig. 1, the example "*not (so good)*" in sentiment analysis illustrates this point. To obtain the polarity of this phrase, we firstly combine the words "*so*" and "*good*", then combine the "*not*" and "*so good*". Specifically, the first combination is a strengthen composition which makes the sentiment polarity stronger, and the second step is a negate composition which negates the positive polarity to negative. This example motivates us to employ multiple composition functions in recursive neural network models. MV-RNN [4] assigns matrices for every words to make compositions specific. However, the number of composition matrices is the same as vocabulary size, which leads to a large number of parameters. Accordingly, it is likely to overfit the training data and difficult to be optimized. Moreover, MV-RNN needs another global matrix to linearly combine composition matrices for phrases, which still makes compositions not specific. Socher *et al.* [5], Hermann and Blunsom [6] propose to use different composition functions according to the combined phrases' syntactic tags extracted from parsing results. But the syntactic categories are not always helpful to guide the compositions. For instance, both "*not good*" and "*very good*" are adv-adj pairs. In the sentiment analysis task, the former is a negator that negates the sentiment polarity, and the other one is a amplifier that increases the polarity strength. This indicates composition functions and com-

Manuscript received June 18, 2015; revised October 19, 2015; accepted November 30, 2015. Date of publication December 17, 2015; date of current version February 11, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant No. 61421003 and in part by the fund of the State Key Lab of Software Development Environment under Grant SKLSDE-2015ZX-05. The work of S. Liu was supported by a Microsoft Research Fund (No. FY15-RES-OPP-112). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Min Zhang.

L. Dong is with the State Key Lab, Software Development Environment, Beihang University, Beijing 100191, China, and also with the Institute for Language, Cognition, and Computation, University of Edinburgh, Edinburgh EH8 9AB, U.K. (e-mail: donglixp@gmail.com).

K. Xu is with State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China (e-mail: kexu@nlsde.buaa.edu.cn).

F. Wei and M. Zhou are with Microsoft Research, Beijing 100080, China (e-mail: fuwei@microsoft.com; mingzhou@microsoft.com).

S. Liu is with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: shixia@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2015.2509257

position selection methods should be jointly learnt from data to utilize the supervision of specific tasks. In order to overcome these shortcomings and make compositions specific, it is better to use a certain number of composition functions and embed the role-sensitive (linguistic and semantic) information into word vectors to adaptively select these compositions, rather than determining them by concrete words or linguistic tags.

In this article, we introduce a novel Adaptive Multi-Compositionality (AdaMC) method for RNNMs. AdaMC consists of more than one composition functions and adaptively selects them depending on input vectors. The model learns to embed the semantic categories of words into their corresponding word vectors and uses them to choose these composition functions adaptively. In particular, we propose a parametrization method to compute the probability distribution of functions for given combined vectors. A hyperparameter is used to model adaptive preferences over the different composition functions. Also, we depict three special cases of AdaMC. By adjusting this hyperparameter, there is a continuous transition between these three special cases. Moreover, all these composition functions and how to select them are jointly learnt from supervision, instead of choosing them heuristically or manually. Hence, task-specific composition information can be embedded into representations and used in the task. Extensive experiments are conducted for the fine-grained sentiment classification task on Stanford Sentiment Treebank, and the semantic relation classification task in SemEval-2010. Experimental results demonstrate that our approach significantly improves the baseline methods, and yields state-of-the-art performances for these tasks.

The main technical contributions of our work are as follows:

- An Adaptive Multi-Compositionality layer that jointly learns multiple composition functions and the adaptive selection method from supervision of specific tasks;
- A parametrization method to make a trade-off between different adaptive preferences and calculate the distribution of composition functions for given vectors;
- A set of experiments on Stanford Sentiment Treebank and semantic relation classification to show that AdaMC outperforms the baseline approaches and improves the state-of-the-art recursive neural network models.

This article presents an extended version of [7]. Specifically, we rewrite Section II to better compare with related work, and provide more technical details about our method. Moreover, we add experiments on investigating the use of syntactic tags to select composition functions, and provide analyses about the multiple composition functions and adaptive selection process in Section VI. We further apply our method to semantic relation classification in Section VII.

II. Related Work

A. Semantic Composition

Semantic composition has attracted extensive attention in vector based semantic models, and a variety of composition functions have been proposed in previous works. Without losing generality, we take the composition of two words as an example. Let a and b be two words, represented by the vectors $\mathbf{a}$ and $\mathbf{b}$. The goal of semantic composition is to compute a vector $\mathbf{c}$ to represent the phrase ab.

To achieve this goal, [9] use the average of $\mathbf{a}$ and $\mathbf{b}$ to obtain the representation for ab, which ignores word order. Mitchell and Lapata [10] suggest using weighted addition $\mathbf{c} = A\mathbf{a} + B\mathbf{b}$ and tuning the weight matrices A and B. If the A and B are unequal, it is sensitive to the word orders. They further compare with the element-wise multiplication composition ($\mathbf{c} = \mathbf{a} \odot \mathbf{b}$, where $c_i = a_i \cdot b_i$), which multiplies the different semantic dimensions instead of addition operation. Experimental results indicate this approach achieves better performances than addition methods.

Another operation is the tensor product [11], [12], [13], such as the outer product, for semantic composition. The outer product of two vectors $\mathbf{a}, \mathbf{b}$ produces a matrix $\mathbf{C}$. If we consider a phrase with three words, the composition result is a third-order tensor. It makes the representations become exponentially larger, which limits its applications.

Besides representing all the words as vectors, Baroni and Zamparelli [14] represent nouns as vectors, and adjectives as matrices. Then the authors apply matrix-by-vector multiplication for the adj-noun pair composition. The composition result is $\mathbf{c} = A\mathbf{b}$, where A is a matrix for adjective a to modify the noun vector $\mathbf{b}$. Rudolph and Giesbrecht [15], Yessenalina and Cardie [16] use matrices to represent the phrases and define the composition functions as matrix multiplication.

The vector representations for individual words have achieved impressive results [17]. However, it is still a challenge to combine these word vectors and figure out the representations for longer phrases, sentences, and beyond. Further, most of previous works evaluate the composition methods by comparing the similarity between short phrases (e.g., adj-noun word pairs). And their settings are unsupervised instead of using the supervision from specific tasks. For example, the "*very good*" and "*very bad*" are regarded as similar phrases, which sometimes is not feasible for the specific tasks (e.g., sentiment analysis). Moreover, these methods obtain word vectors from co-occurrence matrix [8]. In contrast, our method learns the *distributed representations* from the specific tasks instead of being based on this hypothesis.

B. Composition in Neural Network Models

Recently, there are some efforts to integrate semantic composition into neural networks to build deep models for NLP tasks. These deep models learn the distributed representations for words, and use composition functions to obtain the representation vectors for phrases and sentences. The vectors are then used to predict for specific tasks.

Socher *et al.* [2] learn a matrix W, which assigns different weights for every dimensions of $\mathbf{a}$ and $\mathbf{b}$, to combine the vectors $\mathbf{a}, \mathbf{b}$. Socher *et al.* [3] Yu *et al.* [18] use a tensor layer to model the interactions between different dimensions of the combined vectors. The tensor layer produces a bilinear form for the vectors. We describe these two methods in Section III. Socher *et al.* [4] assign matrix-vector pairs $(\mathbf{A}, \mathbf{a})$ and $(\mathbf{B}, \mathbf{b})$ for words a and b, where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{D \times D}$ are parameter matrices, and $\mathbf{a}, \mathbf{b}$ are word representations. The matrices are used to modify the

vectors. When two phrases are combined, the composition result is $\mathbf{c} = \mathbf{W}[\begin{matrix}\mathbf{Ba}\\ \mathbf{Ab}\end{matrix}]$, where $\mathbf{W}$ is a global linear mapping matrix. In addition, this model needs another matrix $\mathbf{O}$ to obtain the operation matrix for phrase c, i.e., $\mathbf{C} = \mathbf{O}[\begin{matrix}\mathbf{A}\\ \mathbf{B}\end{matrix}]$. It assigns operation matrices for every words, which makes the number of parameter grow as the vocabulary size increases. In other words, if we have $|V|$ words, we need $\Theta(|V|D^2)$ parameters for these matrices.

Some works explore utilizing parsers or external resources to improve the compositions in neural models. Grefenstette and Sadrzadeh [19] present a categorical compositional model, which employs formal semantics (grammatical structure) to guide the composition sequence. Hermann and Blunsom [6] use the parsing results based on Combinatory Categorial Grammar (CCG) to guide the semantic composition. Socher *et al.* [5] propose to compute the composition vectors depending on two phrases' syntactic categories extracted from a Probabilistic Context-Free Grammar (PCFG). Our work is significantly different from them. We do not employ any syntactic categories that are assumed to have been obtained by existing parsers. In addition, sometimes the syntactic categories are not helpful to guide the compositions for some tasks. For instance, both "*not good*" and "*very good*" are adv-adj pairs. However, the former is a negation composition which negates the polarity strength, and the other one is a amplifier in the sentiment analysis task. Our proposed method learns to select the composition functions depending on the current vectors (adaptively). Our approach learns the semantic categories from data and apply them to conduct the compositions. Moreover, our model can also leverage these syntactic knowledge in a unified way by regarding them as features, instead of using them heuristically or manually.

III. Background: Recursive Neural Network Models

The Recursive Neural Network Models [2], [20] are one family of neural models, and they have achieved some promising results for the natural language processing tasks. They represent phrases and words as D-dimensional vectors and employ matrices or tensors as operators to perform compositions. The models perform compositions based on binary trees, and obtain vector representations in a bottom-up way. They represent words as vectors and use semantic compositions to compute the vector representations with the same dimension for phrases and sentences. Unlike unsupervisedly obtaining the word vectors from document-word occurrence matrix, the word vectors in the leaf nodes are regarded as the parameters and will be updated according to supervision.

The constituent parse trees are used to decide the order of compositions. Parsing trees are binarized so that each composition consists of only two vectors. Other parsing structures [6] can also be used, while determining the composition structure is not our main point in this article.

Specifically, phrase vectors are computed by compositions of their child vectors. The vector of node i is calculated via:

$$\mathbf{v}^i = f\left(g\left(\mathbf{v}_l^i, \mathbf{v}_r^i\right)\right) \tag{1}$$

where $\mathbf{v}_l^i, \mathbf{v}_r^i$ are the vectors of its left child and right child, g is the composition function, and f is the nonlinearity function

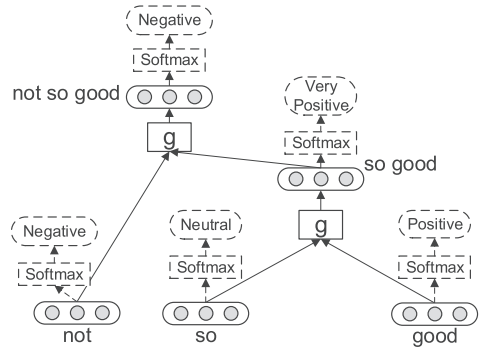

Fig. 2. Composition process for "*not so good*" in recursive neural network models, where g is a global composition function.

(such as tanh, sigmoid, softsign, etc.). The function g project two D-dimensional vectors to a D-dimensional vector, and f apply the nonlinearity for each element. Notably, the works related to semantic composition in the distributional semantics field often do not employ the nonlinearity functions.

As illustrated in Fig. 2, it is a subtree for the phrase "*not so good*". The representation of "*so good*" is calculated by $f(g(\mathbf{v}^{so}, \mathbf{v}^{good}))$. The representation of tri-gram "*not so good*" is recursively obtained by $f(g(\mathbf{v}^{not}, \mathbf{v}^{so\ good}))$.

The learnt representations are then fed into a $\mathrm{softmax}$ classifier to predict labels of nodes. The $\mathrm{softmax}$ layer is used as a standard component, as shown in Fig. 2. The $\mathrm{softmax}$ layer outputs the probability distribution of K classes (such as *positive*, *negative* and *neutral* in the sentiment classification task) for a given input. To be more specific, the prediction distribution of node i is calculated by $\mathbf{y}^i = \mathrm{softmax}(\mathbf{U}\mathbf{v}^i)$ where $\mathbf{U} \in \mathbb{R}^{K \times D}$ is the classification matrix which stacks up $\mathbf{U}_1^T \ldots \mathbf{U}_K^T$ in rows, $\mathbf{v}^i$ is the vector representation of node i, and $\mathbf{y}^i$ is the prediction distribution for node i. The $\frac{1}{\sum_j \exp\{\mathbf{U}_j^T \mathbf{v}^i\}}$ normalizes the distribution and makes the output vector sum to one.

The main differences between recursive neural network models lie in the design of composition functions ($g(\mathbf{v}_l^i, \mathbf{v}_r^i)$ as shown in Equation (1)). We describe two mainstream models and their composition functions, which are also used as our basic units.

A. RNN: Recursive Neural Network

RNN [2], [20] is a standard member of recursive neural network models. Every dimension of parent vector is calculated by weighted **linear** combination of child vectors' dimensions. The vector representation of node i is obtained via:

$$\mathbf{v}^i = f\left(g\left(\mathbf{v}_l^i, \mathbf{v}_r^i\right)\right) = f\left(\underbrace{\mathbf{W}\begin{bmatrix}\mathbf{v}_l^i\\ \mathbf{v}_r^i\end{bmatrix}}_{linear} + \underbrace{\mathbf{b}}_{bias}\right) \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{D \times 2D}$ is the composition matrix, $\mathbf{b}$ is the bias vector, $\mathbf{v}_l^i, \mathbf{v}_r^i$ are the left and right child vectors respectively, and f is the nonlinearity function. The dimension of $\mathbf{v}_i$ is the same as its child vectors, and it is recursively used in the next composition. Notably, the composition matrix and word vectors are all parameters learnt from data.

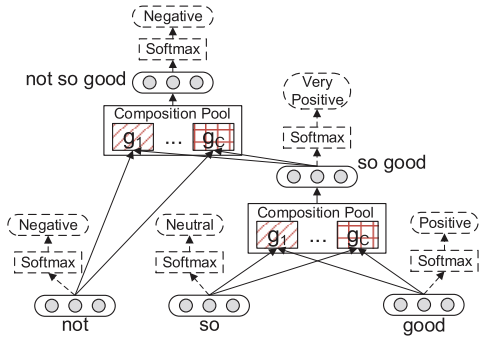

Fig. 3. The composition pool consists of multiple composition functions. It selects the functions depending on the input child vectors, and produces the composition result using more than one composition function.

B. RNTN: Recursive Neural Tensor Network

RNTN [3] uses more parameters and a more powerful composition function than RNN. The main idea of RNTN is to employ tensors to model **interactions** between dimensions. The vector representation of node i is computed via:

$$\begin{aligned}\mathbf{v}^i &= f\left(g\left(\mathbf{v}_l^i, \mathbf{v}_r^i\right)\right) \\ &= f\left(\underbrace{\begin{bmatrix}\mathbf{v}_l^i \\ \mathbf{v}_r^i\end{bmatrix}^T \mathbf{T}^{[1:D]} \begin{bmatrix}\mathbf{v}_l^i \\ \mathbf{v}_r^i\end{bmatrix}}_{interaction} + \underbrace{\mathbf{W}\begin{bmatrix}\mathbf{v}_l^i \\ \mathbf{v}_r^i\end{bmatrix}}_{linear} + \underbrace{\mathbf{b}}_{bias}\right)\end{aligned} \tag{3}$$

where $\mathbf{T}^{[1:D]}$ is a tensor, $\mathbf{T}^{[d]} \in \mathbb{R}^{2D\times 2D}$ is the d-th slice of $\mathbf{T}^{[1:D]}$, $\mathbf{W} \in \mathbb{R}^{D\times 2D}$ is a linear composition matrix, $\mathbf{b}$ is the bias vector, and f is the nonlinearity function. Tensor product obtains a D-dimensional vector which defines multiple bilinear forms. The results are weighted linear combination of interactions between dimensions of $\mathbf{v}_l^i, \mathbf{v}_r^i$. When all the elements of $\mathbf{T}^{[1:D]}$ are zero, the model is the same as RNN. Experimental results [3] indicate that RNTN achieves better performances than RNN for the sentiment classification task.

IV. Adaptive Multi-Compositionality for Recursive Neural Network Models

We illustrate the main idea of proposed Adaptive Multi-Compositionality method in Fig. 3. We use a *composition pool* which consists of C composition functions $\{g_1, \ldots, g_C\}$. To obtain the vector of "*so good*", we firstly feed its child vectors ($\mathbf{v}^{so}$ and $\mathbf{v}^{good}$) into a classifier and get the probability $P(g_h|\mathbf{v}^{so}, \mathbf{v}^{good})$ for using each composition function g_h. Intuitively, we should choose composition functions which strengthen the polarity of "*good*", and predict the "*so good*" as very positive. Similarly, we compute the probability $P(g_h|\mathbf{v}^{not}, \mathbf{v}^{so\ good})$ for every composition function. Ideally, we should select negation composition functions to obtain $\mathbf{v}^{not\ so\ good}$, and the sentiment polarity should be negated. As shown in this example, it is more reasonable to use multiple composition functions than finding a complex global composition function for recursive neural network models.

Generally, we define the composition result $\mathbf{v}^i$ as:

$$\mathbf{v}^i = f\left(\sum_{h=1}^{C} P\left(g_h|\mathbf{v}_l^i, \mathbf{v}_r^i\right) g_h\left(\mathbf{v}_l^i, \mathbf{v}_r^i\right)\right) \tag{4}$$

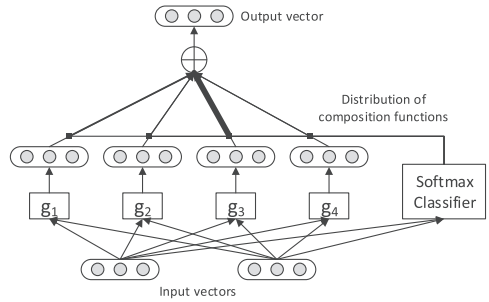

Fig. 4. AdaMC consists of multiple composition functions, and it uses a softmax classifier to compute the distribution of them.

where f is the nonlinearity function, $g_1, \ldots, g_C$ are the composition functions, and $P(g_h|\mathbf{v}_l^i, \mathbf{v}_r^i)$ is the probability of employing g_h given the child vectors $\mathbf{v}_l^i, \mathbf{v}_r^i$. For composition functions, we use the same forms as in RNN (Equation (2)) and RNTN (Equation (3)). Considering compositions of the vector $\mathbf{a}$ and other vectors, the standard composition functions always use the same composition parameters for them, while our method has the capability of adaptively selecting different composition types depending on the roles of input vectors. Moreover, the standard composition is a special case of the proposed method if we always select a specific composition function or set the number of composition functions to one.

The key point is how to select them properly depending on the child vectors, i.e., how to define $P(g_h|\mathbf{v}_l^i, \mathbf{v}_r^i)$. We define a parametrization approach (named AdaMC), and show three special cases of it. By adjusting the parameter of AdaMC, there is a continuous transition between the special cases.

A. AdaMC: Adaptive Multi-Compositionality

As shown in Fig. 4, AdaMC uses the combined word vectors $\mathbf{v}_l^i, \mathbf{v}_r^i$ as features for node i, and feeds them into a softmax classifier to compute the probability distribution of composition functions via:

$$\begin{bmatrix} P\left(g_1|\mathbf{v}_l^i, \mathbf{v}_r^i\right) \\ \vdots \\ P\left(g_C|\mathbf{v}_l^i, \mathbf{v}_r^i\right) \end{bmatrix} = \mathrm{softmax}\left(\beta\mathbf{S}\begin{bmatrix}\mathbf{v}_l^i \\ \mathbf{v}_r^i\end{bmatrix}\right) \tag{5}$$

where β is a hyperparameter, $\mathbf{S} \in \mathbb{R}^{C\times 2D}$ is the parameter matrix used to determine which composition function we use, and $\mathbf{v}_l^i, \mathbf{v}_r^i$ are the left and right child vectors.

When $\beta = 1$, it is the same as softmax function; when $\beta = 0$, the function produces an uniform distribution; when $\beta \to +\infty$, this function only activates the dimension with maximum weight, and sets its probability to 1. If we adjust the $\beta \in [0, +\infty)$, there is a transition between averaging the different composition results and outputting the result with maximum probability. It allows us determine how to adaptively combine these composition functions.

B. Avg-AdaMC: Average AdaMC

Avg-AdaMC averages the composition results, which is a special case of AdaMC ($\beta = 0$). The probability of using g_h:

$$P\left(g_h|\mathbf{v}_l^i, \mathbf{v}_r^i\right) = \frac{1}{C} \tag{6}$$

where $h = 1, \ldots, C$, and C is the number of composition functions. This setting treats all functions equally and does not adaptively select them depending on the combined vectors.

C. Weighted-AdaMC: Weighted Average AdaMC

One special case of AdaMC is setting $\beta = 1$. It uses the softmax probability to perform a weighted average.

D. Max-AdaMC: Max Output AdaMC

If we set $\beta \to +\infty$ in AdaMC, it is a greedy selection algorithm and only outputs the composition result with maximum weight. The probability of using composition function g_h is computed via:

$$P\left(g_h|\mathbf{v}_l^i, \mathbf{v}_r^i\right) = \begin{cases} 1, \mathbf{S}_h^T \begin{bmatrix} \mathbf{v}_l^i \\ \mathbf{v}_r^i \end{bmatrix} > \mathbf{S}_j^T \begin{bmatrix} \mathbf{v}_l^i \\ \mathbf{v}_r^i \end{bmatrix} \text{ for any } j \neq h \\ 0, \text{otherwise} \end{cases} \tag{7}$$

where $\mathbf{S}_1^T \ldots \mathbf{S}_C^T$ are the row vectors of $\mathbf{S}$.

V. Model Training

In this section, we describe how to estimate parameters for classification tasks. The objective function is minimizing the cross-entropy error with L_2-regularizer. The back-propagation algorithm [21] is employed to jointly learn the composition functions and parameters used for adaptive selection. Further details can be found in the supporting document.

A. Objective Function

We use the softmax classifier to predict the probabilities for classes, and compare the distribution with ground truth. We define the target vector $\mathbf{t}^i$ for node i, which is a binary vector. If the desired output class is k, we set $\mathbf{t}_k^i$ to 1 and the others to 0. Our goal is to minimize the cross-entropy error between the predicted distribution $\mathbf{y}^i$ and target distribution $\mathbf{t}^i$. For each sentence, the errors of all nodes are added up. If only the root nodes have the gold labels, we ignore the errors for the other nodes. The objective function is defined as:

$$\min_{\Theta} \mathcal{O}(\Theta) = -\sum_i \sum_j \mathbf{t}_j^i \log \mathbf{y}_j^i + \sum_{\boldsymbol{\theta} \in \Theta} \lambda_{\boldsymbol{\theta}} \|\boldsymbol{\theta}\|_2^2 \tag{8}$$

where Θ represents the set of parameters, $\lambda_{\boldsymbol{\theta}}$ are the hyperparameters, the first term is the cross-entropy error over all the nodes, and the second term is a L_2-regularization penalty.

VI. Experiments for Sentiment Treebank

We use a sentiment treebank with fine-grained sentiment labels to evaluate our approach for the semantic composition in recursive neural network models. Compared with the traditional sentiment classification task, the treebank consists of parse trees, and the polarities of all syntactically plausible phrases are annotated. This annotation scheme is beneficial for learning and analyzing the semantic compositions.

A. Dataset Description

We evaluate the models on the Stanford Sentiment Treebank This corpus contains the labels of syntactically plausible phrases, which allows us to train the compositional models based on parse trees. The treebank is built upon 10,662 critic reviews in Rotten Tomatoes. The Stanford Parser [22] is used to parse all these reviews to parse trees and extract 215,154 phrases. Next, the workers in Amazon Mechanical Turk annotate polarity levels for all these phrases. All the sentiment scales are merged to five categories: *very negative*, *negative*, *neutral*, *positive*, *very positive*.

B. Experiment Settings

We use the standard dataset splits (train: 8,544, dev: 1,101, test: 2,210) in all experiments. For all these models, we tune the parameters on dev dataset. We use the mini-batch version AdaGrad in our experiments with the batch size between 20 and 30. We employ $f = \tanh$ as the nonlinearity function. To initialize parameters, we randomly sample values from a uniform distribution $\mathcal{U}(-\epsilon, +\epsilon)$, where ϵ is a small value. The composition matrices are initialized by concatenating two identity matrices with randomly sample values. It should be noted that the word vectors are regarded as parameters, and will be updated in the training process.

C. Evaluation

We compare different methods on the Sentiment Treebank in this section to evaluate the effectiveness of our methods. The fine-grained and binary (positive/negative) accuracies at root level are compared.

SVM. Support Vector Machine achieves good performance in sentiment classification [23]. Bag-of-words features are used in experiments.

MNB/bi-MNB. As shown in [24], Multinomial Naïve Bayes often outperforms SVM for sentence-level sentiment classification. The MNB uses uni-gram features, and bi-MNB uses uni-gram and bi-gram features.

VecAvg. This model [9] averages child vectors to obtain the parent vector. It ignores the word order when performing compositions.

RNN/RNTN. Recursive Neural Network [2] and Recursive Neural Tensor Network [3] are described in Section III.

MV-RNN. This model [4] assigns composition matrices for every words. Besides performing compositions to obtain the vector representations, the model uses another global matrix to combine these composition matrices for phrases.

The above baseline results are reported by Socher *et al.* [3].

Tag-RNN/Tag-RNTN. The syntactic categories extracted from Stanford Probabilistic Context-Free Grammar (PCFG) parser are used to guide the compositions [5], [6]. According to the syntactic tags of combined phrases, different composition functions are assigned for them. The size of word vector is set as 15 for Tag-RNN and 10 for Tag-RNTN. Notably, we combine some similar syntactic tags (such as NN, NNP, NNPS, NNS) to one category, which improves performance.

AdaMC. Compared with the original models, AdaMC-RNN and AdaMC-RNTN employ multiple composition functions, and determine composition functions depending on the child vectors in every step. The size of word vectors is set as 25 for AdaMC-RNN and 15 for AdaMC-RNTN. The number of composition functions $C = \{5, 15, 25, 35\}$ is chosen on the

TABLE I
ACCURACY RESULTS ON SENTIMENT TREEBANK

Method	Fine-grained	Binary
SVM	40.7	79.4
MNB	41.0	81.8
bi-MNB	41.9	83.1
VecAvg	32.7	80.1
MV-RNN	44.4	82.9
RNN	43.2	82.4
Tag-RNN	39.1	80.2
Avg-AdaMC-RNN	43.4	84.9
Max-AdaMC-RNN	43.8	85.6
Weighted-AdaMC-RNN	45.4	86.5
AdaMC-RNN	**45.8**	**87.1**
RNTN	45.7	85.4
Tag-RNTN	40.8	83.0
Avg-AdaMC-RNTN	45.7	86.3
Max-AdaMC-RNTN	45.6	86.6
Weighted-AdaMC-RNTN	**46.3**	**88.4**
AdaMC-RNTN	**46.7**	**88.5**

development set. Too small C can not fully utilize capabilities of model, while too large C leads to overfitting. We use 15 composition functions in experiments.

Avg/Max/Weighted-AdaMC. Special cases of AdaMC are used for RNN and RNTN. The number of composition functions and dimension of word vectors are the same as in AdaMC-RNN and AdaMC-RNTN.

Table I shows the evaluation results of different models. SVM and MNB are two effective baselines for sentiment classification [23], [24]. The results of RNN are better than VecAvg. MV-RNN uses more composition matrices and improves the accuracies compared with RNN. Moreover, RNTN, which employs tensors to model the interactions between different semantic dimensions, achieves better results than MV-RNN, RNN and bag-of-words models. This implies that more powerful composition functions help capture complex semantic compositions.

We then compare our method with baselines. First, we apply our approaches for RNN, and there are significant gains of evaluation metrics. Moreover, we find that AdaMC-RNN surpasses MV-RNN. The MV-RNN assigns specific composition matrices for every words. It is easy to overfit training data, because the number of composition matrices is the same as vocabulary size. MV-RNN needs another global matrix to compute the composition matrices for long phrases, which makes the composition non-specific. Furthermore, the fine-grained accuracies of AdaMC-RNN are comparable with RNTN, and the binary accuracies are better than RNTN. The results indicate using multiple composition functions is another good approach to improve the semantic composition besides finding more powerful composition functions. We also apply our method in RNTN, and obtain better performances. This demonstrates our method can boost the performances even if we have used powerful composition functions.

Compared to the Tag-RNN and Tag-RNTN, we find that using syntactic tags to guide compositions does not improve the performance for this task. There are 55 syntactic categories extracted by Stanford Parser. After combining the similar tags to one category, 28 categories are still remained. Hence, 784 ($= 28 * 28$) composition matrices or tensors are employed, which results in having too many parameters. What's more, the syntactic categories do not embed task-specific semantics.

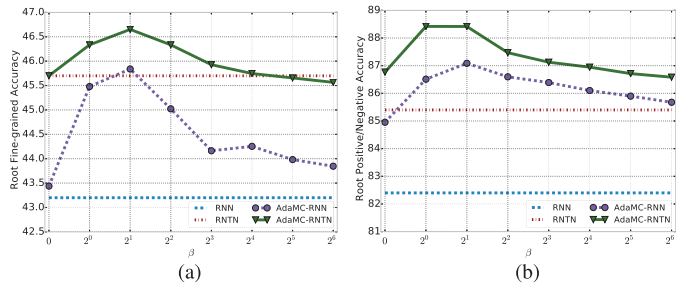

Fig. 5. The curve shows the accuracy for root nodes as $\beta = 0, 2^0, 2^1, \ldots, 2^6$ increases. AdaMC-RNN and AdaMC-RNTN achieve the best results at $\beta = 2^1$, and they both achieve improvements than the RNN and RNTN. (a) Root Fine-grained Accuracy (b) Root Pos/Neg Accuracy.

As the performance is correlated with model size, we further compare the number of parameters for different methods used in the experiments. The model size of RNN, Tag-RNN and AdaMC-RNN is 458 K, 627 K and 475 K respectively, and the size of RNTN, Tag-RNTN and AdaMC-RNTN is 520 K, 3,475 K and 483 K respectively. The MV-RNN has much more parameters than other models because it assigns a word matrix for each word. The experimental results demonstrate that our method does not outperform the baseline methods due to simply increasing the number of parameters.

Finally, we evaluate the special cases ($\beta = 0, 1$, and $\beta \to \infty$) of AdaMC models. The performances of Weighted-AdaMC ($\beta = 1$) are most similar to AdaMC, and are better than Avg-AdaMC ($\beta = 0$) and Max-AdaMC ($\beta \to \infty$). To be specific, the results of Max-AdaMC-RNN surpass Avg-AdaMC-RNN. However, the fine-grained accuracies of Avg-AdaMC-RNTN are slightly better than Max-AdaMC-RNTN, and the positive/negative accuracies are just the opposite. We will further explore the differences between these special cases in next section.

D. *Effects of* β

We compare different β for AdaMC defined in Equation (5) in this section. Different parameter β leads to different composition selection schemes. When $\beta = 1$, the model directly employs the probabilities outputted by the softmax classifier as weights to combine the composition results. The $\beta = 0$ makes the probabilities obey an uniform distribution, while $\beta \to \infty$ results in a maximum probability selection algorithm.

As demonstrated in Fig. 5, the overall conclusion is that the optimal β tends to be between the setting of Weighted-AdaMC and Max-AdaMC. Both the AdaMC-RNN and AdaMC-RNTN achieve the best root fine-grained and positive/negative accuracies at $\beta = 2$, and they have a similar trend. Specifically, the Weighted-AdaMC ($\beta = 1$) performs better than Avg-AdaMC ($\beta = 0$) and Max-AdaMC ($\beta \to \infty$). It indicates that adaptive (role-sensitive) compositionality selection is useful to model the compositions. The phrase-level results also follow the similar trend on this dataset.

The Avg-AdaMC does not consider role-sensitive information, while Max-AdaMC does not use multiple composition

TABLE II
NEIGHBORS OF PHRASES BY QUERYING COMPOSITION SELECTION VECTORS

really bad	very bad / only dull / much bad / extremely bad / (all that) bad
(is n't) (necessarily bad)	(is n't) (painfully bad) / not mean-spirited / not (too slow) / not well-acted / (have otherwise) (been bland)
great (Broadway play)	great (cinematic innovation) / great subject / great performance / energetic entertainment / great (comedy filmmaker)
(arty and) jazzy	(Smart and) fun / (verve and) fun / (unique and) entertaining / (gentle and) engrossing / (warmth and) humor

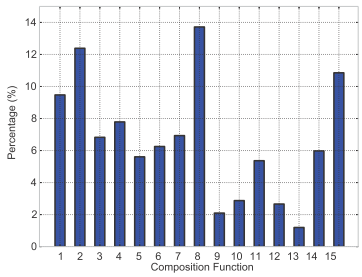

Fig. 6. For the compositions, we show the distribution of composition functions which have largest adaptive selection probabilities. It implies that adaptive composition selections are not biased towards few ones.

functions to get smoothed results. Weighted-AdaMC employs the probabilities obtained by softmax classifier to make trade-offs between them, hence it obtains better performances. Based on this observation, we introduce the hyperparameter β in AdaMC to adjust the effects of these two perspectives.

E. Adaptive Composition Analysis

In order to analyze the adaptive composition selection, we compute probability distribution (as in Equation (5)) for every composition, i.e., $[P(g_1|\mathbf{a},\mathbf{b})\dots P(g_C|\mathbf{a},\mathbf{b})]^T$ where $\mathbf{a},\mathbf{b}$ are child vectors. The distribution vectors are used to compute the similarities between composition pairs. If the distributions of two pairs are more similar, the model obtains their composition results in more similar manner. In this section, we show some composition examples and their neighbors.

As demonstrated in Table II, we query some composition examples and employ cosine similarity as the similarity metric. To be specific, "*really bad*" is a strengthen composition which makes the sentiment polarity stronger, and the most similar compositions are of the same type. Notably, we notice that the bi-gram "*all that*" is detected as an intensification indicator. The second example is a negation composition. "*(have otherwise) (been bland)*" is regarded as a similar composition, which illustrates the combination of "*have*" and "*otherwise*" keeps the negation semantics of "*otherwise*". The third case and the similar compositions are combinations of a sentiment adjective word and noun phrase. This demonstrates our model embeds some word-category information in word vectors to select composition functions. The last example is two sentiment words connected by "*and*". The results illustrate our model recognize this kind of conjunction which joins two non-contrasting items. From all these cases, we find the compositions which are of similar types are closer, and our method learns to distinguish these different composition types according to the supervision of specific tasks.

In addition, for every composition function, we count the number of times when its adaptive selection probabilities are the largest. Fig. 6 shows the distribution for these composition functions. It illustrates that all composition functions contribute to the compositions, instead of only few functions which play dominated roles.

F. Composition Matrices Visualization

We visualize the composition matrices of AdaMC-RNN and RNN to illustrate the behaviors of model. The number of composition functions is set as 15 in our experiments. We use grids to represent the elements of matrices, and the colors indicate their values.

Fig. 7(a) shows the composition matrices of AdaMC-RNN, and Fig. 7(b) shows the composition matrix of RNN. The left and right blocks of a matrix are used to multiply the left and right combined vectors, respectively. It meets our intuitions that most of diagonal elements are positive to combine the dimensional semantics of two phrases. Compared to RNN, we find that AdaMC-RNN learns more diverse composition matrices which improves the performance. If a diagonal is stronger than the other one, it means this composition matrix assigns more weights and give more importances for this part of combined vectors. For instance, to compute the representation of "*a masterpiece*", more semantics of "*masterpiece*" should be remained by composition functions. Besides the diagonals of AdaMC-RNN, the other elements of two blocks are also more diverse than RNN. Because RNN has only one composition matrix for all pairs, it tends to average the combined vectors without using specific composition functions for phrases. In addition, the diverse blocks imply that AdaMC-RNN captures varied kinds of linear interactions between dimensions, which result in specific compositions.

G. Contrast Compositionality

The contrast conjunction [25] is a joiner to express oppositions. For instance, the sentiment polarity of "*It maybe good, but I still do not like it.*" is classified to negative. To evaluate the performances for modeling contrast compositionality, we extract a subset from test set which includes the sentences with "X *but* Y" structure. We only keep the sentences whose polarities of X and Y are different (including neutral). After filtering, we obtain 131 cases for experiments. We then perform classification on this dataset. We say a case is classified correctly, if the polarities of nodes "X *but* Y", "X", and "Y" are all classified to the correct classes.

As shown in Table III, we find that our method improves the performances for handling contrast compositionality. First of all, we give a glance at the baselines. The bi-MNB is a bag-of-words approach, and it has not the abilities to model the contrast compositionality. The recursive neural network models capture this linguistic phenomenon by word vectors and composition functions. The accuracy results of RNN and RNTN are better than bi-NB. After using AdaMC for RNN and RNTN, the performances become better. The accuracies

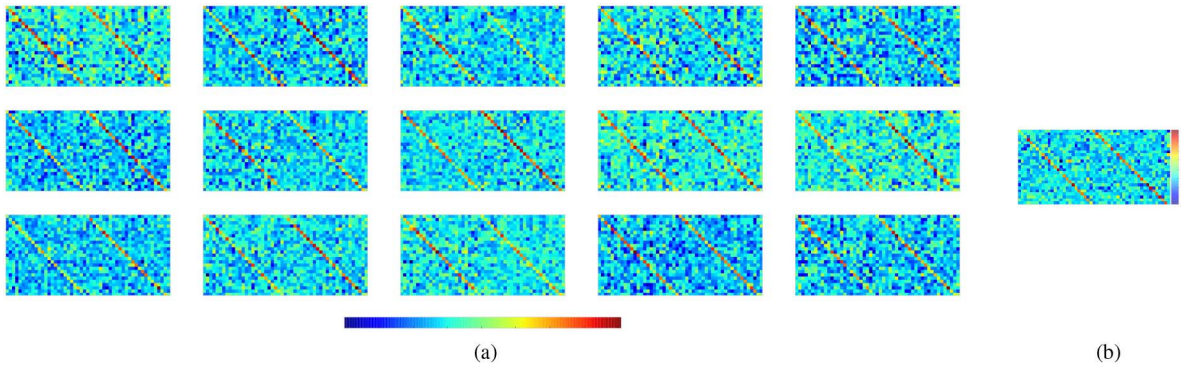

Fig. 7. The composition matrices of AdaMC-RNN and RNN are visualized. Every grid is an element of matrices. If the color is more warm, its value is larger. Best viewed in color. (a) Composition Matrices of AdaMC-RNN (b) Composition Matrix of RNN.

TABLE III
ACCURACY RESULTS ON CONTRAST SENTENCES ("X *BUT* Y")

Method	Accuracy
bi-MNB	27
RNN	36
MV-RNN	37
RNTN	41
AdaMC-RNN	44
AdaMC-RNTN	46

improve 17% and 19% respectively than bi-MNB. The results indicate that our method can more accurately capture the contrast compositionality.

VII. EXPERIMENTS FOR SEMANTIC RELATION CLASSIFICATION

To verify effectiveness of AdaMC for the other tasks, we evaluate our approach for semantic relation classification task. This task is to predict which of the semantic relation labels to apply for two tagged entities in a sentence. For instance, in the sentence "*People have been moving back into downtown*.", we want to predict the word pair "*people*" and "*downtown*" as a *Entity-Destination* semantic relation. The classification of semantic relations have many applications, such as document summarization, and information extraction. The previous state-of-the-art approach is to use Support Vector Machines with feature engineering. We demonstrate that we can achieve better results by employing our method (AdaMC) with only three additional lightweight features.

To predict the semantic relations using recursive neural network models, we firstly extract the subtree which dominates predicted word pair from parse results [4]. The parse trees are computed by Stanford Parser [22] as in Section VI. Then we perform compositions to obtain the vector representation of subtree, and classify the semantic relations at the root node using softmax classifier. The basic configurations of experiments is similar to the sentiment treebank classification task. The nonlinearity function is also used due to better performances. The word vectors are initialized by the pre-trained 50-dimensional word embedding [26], which are updated in the training process. The composition matrices are initialized by concatenating two identity matrices plus randomly sample values, and the other parameters are randomly initialized with small values.

A. Dataset Description

We employ the dataset of SemEval-2010 Task #8 (Multi-Way Classification of Semantic Relations Between Pairs of Nominals). This dataset consists of 8,000 sentences for training and 2,717 sentences for testing. For both relation directions, the dataset consists of nine relation types. And there is a *Other* class. Hence, the dataset has 19 classes for semantic relations. Our task is to classify the word pairs with considering the relation directions.

B. Evaluation

We use the same evaluation framework of SemEval-2010. We evaluate different methods using the macro-averaged F1-score over the 18 semantic relation types apart from *Other* class, and semantic relation directions are considered. We compare the top methods participated in SemEval-2010, which are summarized by Hendrickx *et al.* [27]. Most of them use apply feature engineering and rely on external linguistic resources to extract features.

SVM. Support Vector Machines with kinds of features achieve best result in the SemEval-2010.

MaxEnt. Maximum Entropy Classifier is widely used for a wide variety of text classification tasks.

RNN. Recursive Neural Network [2] consists of a global linear composition function to combine the child vectors. The dimension of word vectors is 50 in experiments, and they are pre-trained by the model of [26].

MV-RNN. This model [4] assigns composition matrices for every words. Besides performing compositions to obtain vector representations, the model uses another global matrix to combine these composition matrices for phrases. To reduce the number of parameters, the composition matrices are represented by low-rank plus diagonal approximations. The approximation rank is 3, and the word vectors are initialized by the pre-trained 50-dimensional word embedding [26].

The above baseline results are reported by Socher *et al.* [4].

AdaMC-RNN. Compared with the RNN, AdaMC-RNN employs multiple composition functions and selects them depending on the child vectors in every step. The number of composition functions is 5 in experiments. The same pre-trained word vectors as in RNN and MV-RNN are used, and the other parameters are randomly initialized.

TABLE IV
EVALUATION RESULTS FOR PREDICTING SEMANTIC RELATIONS

Model	Features	F1 score
MaxEnt	POS, WordNet, Google n-grams, thesauri, morphological features	77.6
SVM	POS, WordNet, Google n-grams, noun compound system, thesauri, morphological features	77.6
SVM	POS, WordNet, Google n-grams, morphological features, PropBank, FrameNet, dependency parse, paraphrases, TextRunner, etc.	82.2
RNN	-	74.8
MV-RNN	-	79.1
AdaMC-RNN	-	80.6
RNN	POS, WordNet, NER	77.6
AdaMC-RNN	POS, WordNet, NER	**82.4**

As illustrated in Table IV, AdaMC-RNN without employing any manually designed features outperforms most of other methods. By using three lightweight features, the performance of our method increases and it achieves the best result. Specifically, the performances of feature engineering baselines increase as the number of feature sets increases. Among these baseline methods, the macro-averaged F1-score of the winner system in SemEval-2010 is 82.2% [27], which employs lots of manually designed features and external resources.

To start with, we evaluate the RNN, MV-RNN, and AdaMC-RNN without using any additional features. The result of RNN is 74.8%, and the MV-RNN reaches 79.1%. The F1-score of our method AdaMC-RNN reaches 80.6%, which improves 5.8% and 1.5% over RNN and MV-RNN respectively. This indicates that adaptively employing multiple composition functions is useful to achieve better performances. Compared with the feature engineering methods, AdaMC-RNN, without employing hand-crafted features, achieves better result than the system which ranks the second place in SemEval evaluation contest. We further employ three lightweight features computed by the tool of Ciaramita and Altun [28]. With the help of them, our method outperforms all the other baseline methods including the best method with extensive manually designed features.

VIII. CONCLUSION

We propose an Adaptive Multi-Compositionality (AdaMC) method for recursive neural network models. AdaMC uses more than one composition functions and adaptively selects them depending on the input vectors. The composition functions and their distribution are jointly learnt from the supervision of specific tasks. We integrate AdaMC into existing popular recursive neural network models (e.g., RNN and RNTN) and conduct experiments for sentiment analysis task and semantic composition task. Experimental results show that AdaMC performs better than the baselines. Based on the model learnt from Sentiment Treebank, we further compare the distribution similarities of composition functions for phrase pairs, and analyze the learnt multiple composition matrices. The results verify the effectiveness of AdaMC on modeling and leveraging semantic categories of words and phrases in the process of composition.

REFERENCES

[1] P. D. Turney, "Domain and function: A dual-space model of semantic relations and compositions," *J. Artif. Intell. Res.*, vol. 44, pp. 533–585, 2012.

[2] R. Socher, C. C. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *Proc. ICML*, 2011.

[3] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP*, 2013, pp. 1631–1642.

[4] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proc. EMNLP-CoNLL*, 2012, pp. 1201–1211.

[5] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng, "Parsing with compositional vector grammars," in *Proc. ACL*, 2013.

[6] K. M. Hermann and P. Blunsom, "The role of syntax in vector space models of compositional semantics," in *Proc. ACL*, 2013, pp. 894–904.

[7] L. Dong, F. Wei, M. Zhou, and K. Xu, "Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis," in *Proc 28th AAAI Conf. Artif. Intell.*, 2014, AAAI.

[8] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *J. Artif. Intell. Res.*, vol. 37, no. 1, pp. 141–188, Jan. 2010.

[9] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge," *Psychol. Rev.*, pp. 211–240, 1997.

[10] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Sci.*, vol. 34, no. 8, pp. 1388–1429, 2010.

[11] P. Smolensky, "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artif. Intell.*, vol. 46, pp. 159–216, 1990.

[12] S. Clark, B. Coecke, and M. Sadrzadeh, "A compositional distributional model of meaning," in *Proc. 2nd Quantum Interact. Symp.*, 2008, pp. 133–140.

[13] D. Widdows, "Semantic vector products: Some initial investigations," in *Proc. 2nd AAAI Symp. Quantum Interact.*, 2008, vol. 26.

[14] M. Baroni and R. Zamparelli, "Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space," in *Proc. Empir. Meth. Nat. Lang. Process.*, 2010, pp. 1183–1193, ser. EMNLP'10.

[15] S. Rudolph and E. Giesbrecht, "Compositional matrix-space models of language," in *Proc. ACL*, Stroudsburg, PA, USA, 2010, pp. 907–916.

[16] A. Yessenalina and C. Cardie, "Compositional matrix-space models for sentiment analysis," in *Proc. Empir. Meth. Nat. Lang. Process.*, 2011, pp. 172–182.

[17] R. Rapp, "Word sense discovery based on sense descriptor dissimilarity," in *Proc. 9th Mach. Transl. Summit.*, 2003, pp. 315–322.

[18] D. Yu, L. Deng, and F. Seide, "The deep tensor neural network with applications to large vocabulary speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 21, no. 2, pp. 388–396, Feb. 2013.

[19] E. Grefenstette and M. Sadrzadeh, "Experimental support for a categorical compositional distributional model of meaning," in *Proc. Empir. Meth. Nat. Lang. Process.*, 2011, pp. 1394–1404.

[20] C. Goller and A. Küchler, "Learning task-dependent distributed representations by backpropagation through structure," in *Proc. ICNN'96*, 1996, pp. 347–352.

[21] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proc. ACL*, 2003, pp. 423–430.

[23] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. ACL*, 2005, pp. 115–124.

[24] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. ACL*, 2012, pp. 90–94.

[25] D. Blakemore, "Denial and contrast: A relevance theoretic analysis ofbut," *Linguist. Philos.*, vol. 12, no. 1, pp. 15–37, 1989.

[26] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. ICML*, 2008, pp. 160–167.

[27] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. O. Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proc. 5th Int. Workshop Semantic Eval.*, 2010, pp. 33–38.

[28] M. Ciaramita and Y. Altun, "Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger," in *Proc. Empir. Meth. Nat. Lang. Process.*, 2006, pp. 594–602.

Li Dong is a postgraduate student at Beihang University and University of Edinburgh. He received his B.E. degree from the School of Computer Science and Engineering of Beihang University in 2012. His research interests include natural language processing and deep neural network.

Furu Wei received the Ph.D. in June 2009 from Wuhan University. Before that, he received a B.Sc. degree from the same department in 2004. He is a Lead Researcher in Natural Language Computing group at Microsoft Research Asia. Before joining MSRA-NLC in Nov. 2010, he has been a Staff Researcher at IBM Research - China (IBM CRL) since July 2009. His research interests include natural language processing, information retrieval and machine learning.

Ke Xu received his B.E., M.E. and Ph.D. degrees from Beihang University in 1993, 1996 and 2000, respectively. He is a Professor at Beihang University, China. His research interests include algorithm and complexity, data mining and complex networks.

Shixia Liu received a B.S. and M.S. from HIT, a Ph.D. from THU. She is an Associate Professor in the School of Software, Tsinghua University. Her research interests include visual text analytics, visual social analytics, and text mining. Before she joined THU, she worked as a Lead Researcher at MSRA and a Research Staff Member and Research Manager at IBM China Research Lab. She is an associate editor of the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS and is the papers co-chair of IEEE VAST 2016–2017.

Ming Zhou is a Principal Researcher in Natural Language Computing group at Microsoft Research Asia. He works in the areas of machine translation and natural language processing, who has led various projects on NLP, machine translation, text mining and social network. He has authored or co-authored about 100 papers published at top conferences, and has served as area chairs of ACL, UCAI, AAAI, EMNLP, COLING, SIGIR, UCNLP for many times.