Learning Tag Embeddings and Tag-specific Composition Functions in Recursive Neural Network

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu*, Xuan Zhu*, Xiaoyan Zhu
State Key Lab. of Intelligent Technology and Systems, National Lab. for Information Science and Technology, Dept. of Computer Science and Technology, Tsinghua University, Beijing 100084, PR China
*Samsung R&D Institute Beijing, China
qianqiaodecember29@126.com, smxtianbo@gmail.com
aihuang@tsinghua.edu.cn, yang.liu@samsung.com
xuan.zhu@samsung.com, zxy-dcs@tsinghua.edu.cn

Abstract

Recursive neural network is one of the most successful deep learning models for natural language processing due to the compositional nature of text. The model recursively composes the vector of a parent phrase from those of child words or phrases, with a key component named composition function. Although a variety of composition functions have been proposed, the syntactic information has not been fully encoded in the composition process. We propose two models, **T**ag **G**uided RNN (TG-RNN for short) which chooses a composition function according to the part-of-speech tag of a phrase, and **T**ag **E**mbedded RNN/RNTN (TE-RNN/RNTN for short) which learns tag embeddings and then combines tag and word embeddings together. In the fine-grained sentiment classification, experiment results show the proposed models obtain remarkable improvement: TG-RNN/TE-RNN obtain remarkable improvement over baselines, TE-RNTN obtains the second best result among all the top performing models, and all the proposed models have much less parameters/complexity than their counterparts.

1 Introduction

Among a variety of deep learning models for natural language processing, Recursive Neural Network (RNN) may be one of the most popular models. Thanks to the compositional nature of natural text, recursive neural network utilizes the recursive structure of the input such as a phrase or sentence, and has shown to be very effective for many natural language processing tasks including semantic relationship classification (Socher et al., 2012), syntactic parsing (Socher et al., 2013a), sentiment analysis (Socher et al., 2013b), and machine translation (Li et al., 2013).

The key component of RNN and its variants is the composition function: how to compose the vector representation for a longer text from the vector of its child words or phrases. For instance, as shown in Figure 2, the vector of *'is very interesting'* can be composed from the vector of the left node *'is'* and that of the right node *'very interesting'*. It's worth to mention again, the composition process is conducted with the syntactic structure of the text, making RNN more interpretable than other deep learning models.

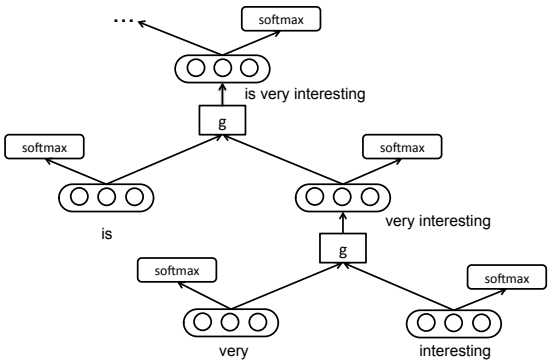

Figure 1: The example process of vector composition in RNN. The vector of node *'very interesting'* is composed from the vectors of node *'very'* and node *'interesting'*. Similarly, the node *'is very interesting'* is composed from the phrase node *'very interesting'* and the word node *'is'* .

There are various attempts to design the composition function in RNN (or related models). In RNN (Socher et al., 2011), a global matrix is used to linearly combine the elements of vectors. In RNTN (Socher et al., 2013b), a global tensor is used to compute the tensor products of dimensions to favor the association between different el-

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1365–1374, Beijing, China, July 26-31, 2015.

ements of the vectors. Sometimes it is challenging to find a single function to model the composition process. As an alternative, multiple composition functions can be used. For instance, in MV-RNN (Socher et al., 2012), different matrices is designed for different words though the model is suffered from too much parameters. In AdaMC RNN/RNTN (Dong et al., 2014), a fixed number of composition functions is linearly combined and the weight for each function is adaptively learned.

In spite of the success of RNN and its variants, the syntactic knowledge of the text is not yet fully employed in these models. Two ideas are motivated by the example shown in Figure 2: **First**, the composition function for the noun phrase *'the movie/NP'* should be different from that for the adjective phrase *'very interesting/ADJP'* since the two phrases are quite syntactically different. More specifically to sentiment analysis, a noun phrase is much less likely to express sentiment than an adjective phrase. There are two notable works mentioned here: (Socher et al., 2013a) presented to combine the parsing and composition processes, but the purpose is for parsing; (Hermann and Blunsom, 2013) designed composition functions according to the combinatory rules and categories in CCG grammar, however, only marginal improvement against Naive Bayes was reported. Our proposed model, *tag guided RNN (TG-RNN)*, is designed to use the syntactic tag of the parent phrase to guide the composition process from the child nodes. As an example, we design a function for composing noun phrase (*NP*) and another one for adjective phrase (*ADJP*). This simple strategy obtains remarkable improvements against strong baselines.

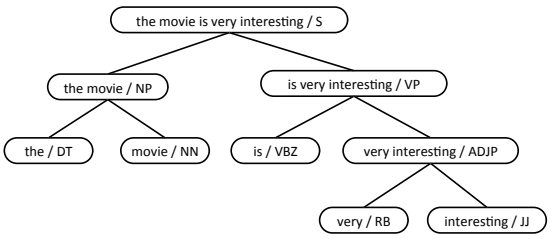

Figure 2: The parse tree for sentence *'The movie is very interesting'* built by Stanford Parser.

Second, when composing the adjective phrase *'very interesting/ADJP'* from the left node *'very/RB'* and the right node *'interesting/JJ'*, the right node is obviously more important than the left one. Furthermore, the right node *'interesting/JJ'* apparently contributes more to sentiment expression. To address this issue, we propose *Tag embedded RNN/RNTN (TE-RNN/RNTN)*, to learn an embedding vector for each word/phrase tag, and concatenate the tag vector with the word/phrase vector as input to the composition function. For instance, we have tag vectors for *DT,NN,RB,JJ,ADJP,NP, etc.* and the tag vectors are then used in composing the parent's vector. The proposed *TE-RNTN* obtain the second best result among all the top performing models but with much less parameters and complexity. To the best of our knowledge, this is the first time that tag embedding is proposed.

To summarize, the contributions of our work are as follows:

- We propose tag-guided composition functions in recursive neural network, TG-RNN. Tag-guided RNN allocates a composition function for a phrase according to the part-of-speech tag of the phrase.
- We propose to learn embedding vectors for part-of-speech tags of words/phrases, and integrate the tag embeddings in RNN and RNTN respectively. The two models, TE-RNN and TE-RNTN, can leverage the syntactic information of child nodes when generating the vector of parent nodes.
- The proposed models are efficient and effective. The scale of the parameters is well controlled. Experimental results on the Stanford Sentiment Treebank corpus show the effectiveness of the models. TE-RNTN obtains the second best result among all publicly reported approaches, but with much less parameters and complexity.

The rest of the paper is structured as follows: in Section 2, we survey related work. In Section 3, we introduce the traditional recursive neural network as background. We present our ideas in Section 4. The experiments are introduced in Section 5. We summarize the work in Section 6.

2 Related Work

Different kinds of representations are used in sentiment analysis. Traditionally, the bag-of-words representations are used for sentiment analysis (Pang and Lee, 2008). To exploit the relationship between words, word co-occurrence (Turney et al., 2010) and syntactic contexts (Padó

and Lapata, 2007) are considered. In order to distinguish antonyms with similar contexts, neural word vectors (Bengio et al., 2003) are proposed and can be learnt in an unsupervised manner. Word2vec (Mikolov et al., 2013a) introduces a simpler network structure making computation more efficiently and makes billions of samples feasible for training.

Semantic composition deals with representing a longer text from its shorter components, which is extensively studied recently. In many previous works, a phrase vector is usually obtained by average (Landauer and Dumais, 1997), addition, element-wise multiplication (Mitchell and Lapata, 2008) or tensor product (Smolensky, 1990) of word vectors. In addition to using vector representations, matrices can also be used to represent phrases and the composition process can be done through matrix multiplication (Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011).

Recursive neural models utilize the recursive structure (usually a parse tree) of a phrase or sentence for semantic composition. In Recursive Neural Network (Socher et al., 2011), the tree with the least reconstruction error is built and the vectors for interior nodes is composed by a global matrix. Matrix-Vector Recursive Neural Network (MV-RNN) (Socher et al., 2012) assigns matrices for every words so that it could capture the relationship between two children. In Recursive Neural Tensor Networks (RNTN) (Socher et al., 2013b), the composition process is performed on a parse tree in which every node is annotated with fine-grained sentiment labels, and a global tensor is used for composition. Adaptive Multi-Compositionality (Dong et al., 2014) uses multiple weighted composition matrices instead of sharing a single matrix.

The employment of syntactic information in RNN is still in its infant. In (Socher et al., 2013a), the part-of-speech tag of child nodes is considered in combining the processes of both composition and parsing. The main purpose is for better parsing by employing RNN, but it is not designed for sentiment analysis. In (Hermann and Blunsom, 2013), the authors designed composition functions according to the combinatory rules and categories in CCG grammar. However, only marginal improvement against Naive Bayes was reported. Unlike (Hermann and Blunsom, 2013), our TG-RNN obtains remarkable improvements against strong baselines, and we are the first to propose tag embedded RNTN which obtains the second best result among all reported approaches.

3 Background: Recursive Neural Models

In recursive neural models, the vector of a longer text (e.g., sentence) is composed from those of its shorter components (e.g., words or phrases). To compose a sentence vector through word/phrase vectors, a binary parse tree has to be built with a parser. The leaf nodes represent words and interior nodes represent phrases. Vectors of interior nodes are computed recursively by composition of child nodes' vectors. Specially, the root vector is regarded as the sentence representation. The composition process is shown in Figure 1.

More formally, vector $v_i \in R^d$ for node i is calculated via:

$$v_i = f(g(v_i^l, v_i^r)) \tag{1}$$

where v_i^l and v_i^r are child vectors, g is a composition function, and f is a nonlinearity function, usually *tanh*. Different recursive neural models mainly differ in composition function. For example, the composition function for RNN is as below:

$$g(v_i^l, v_i^r) = W \begin{bmatrix} v_i^l \\ v_i^r \end{bmatrix} + b \tag{2}$$

where $W \in R^{d \times 2d}$ is a composition matrix and b is a bias vector. And the composition function for RNTN is as follows:

$$g(v_i^l, v_i^r) = \begin{bmatrix} v_i^l \\ v_i^r \end{bmatrix} T^{[1:d]} \begin{bmatrix} v_i^l \\ v_i^r \end{bmatrix} + W \begin{bmatrix} v_i^l \\ v_i^r \end{bmatrix} + b \tag{3}$$

where W and b are defined in the previous model and $T^{[1:d]} \in R^{2d \times 2d \times d}$ is the tensor that defines multiple bilinear forms.

The vectors are used as feature inputs to a softmax classifier. The posterior probability over class labels on a node vector v_i is given by

$$y_i = \text{softmax}(W_s v_i + b_s). \tag{4}$$

The parameters in these models include the word table L, a composition matrix W in RNN, and W and $T^{[1:d]}$ in RNTN, and the classification matrix W_s for the softmax classifier.

4 Incorporating Syntactic Knowledge into Recursive Neural Model

The central idea of the paper is inspired by the fact that words/phrases of different part-of-speech tags play different roles in semantic composition. As discussed in the introduction, a noun phrase (e.g., *a movie/NP*) may be composed different from a verb phrase (e.g., *love movie/VP*). Furthermore, when composing the phrase *a movie/NP*, the two child words, *a/DT* and *movie/NN*, may play different roles in the composition process. Unfortunately, the previous RNN models neglect such syntactic information, though the models do employ the parsing structure of a sentence.

We have two approaches to improve the composition process by leveraging tags on parent nodes and child nodes. One approach is to use different composition matrices for parent nodes with different tags so that the composition process could be guided by phrase type, for example, the matrix for *'NP'* is different from that for *'VP'* . The other approach is to introduce 'tag embedding' for words and phrases, for example, to learn tag vectors for *'NP, VP, ADJP',* etc., and then integrate the tag vectors with the word/phrase vectors during the composition process.

4.1 Tag Guided RNN (TG-RNN)

We propose Tag Guided RNN (TG-RNN) to respect the tag of a parent phrase during the composition process. The model chooses a composition function according to the part-of-speech tag of a phrase. For example, *'the movie'* has tag *NP*, *'very interesting'* has tag *ADJP*, the two phrases have different composition matrices.

More formally, we design composition functions g with a factor of the phrase tag of a parent node. The composition function becomes

$$g(t_i, v_i^l, v_i^r) = g_{t_i}(v_i^l, v_i^r) = W_{t_i} \begin{bmatrix} v_i^l \\ v_i^r \end{bmatrix} + b_{t_i} \quad (5)$$

where t_i is the phrase tag for node i, W_{t_i} and b_{t_i} are the parameters of function g_{t_i}, as defined in Equation 2. In other words, phrase nodes with various tags have their own composition functions such as g_{NP}, g_{VP}, and so on. There are totally k composition function in this model where k is the number of phrase tags. When composing child vectors, a function is chosen from the function pool according to the tag of the parent node. The process is depicted in Figure 3. We term this model Tag guided RNN, TG-RNN for short.

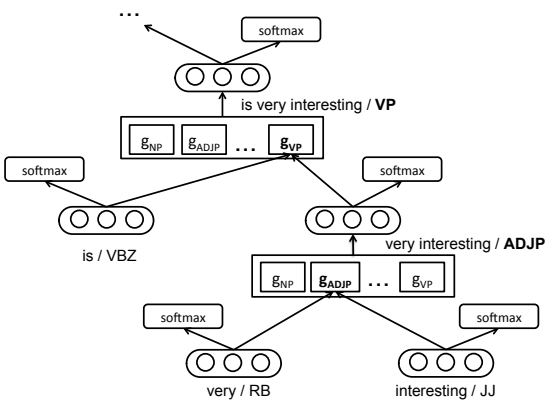

Figure 3: The vector of phrase *'very interesting'* is composed with highlighted g_{ADJP} and *'is very interesting'* with g_{VP}.

But some tags have few occurrences in the corpus. It is hard and meaningless to train composition functions for those infrequent tags. So we simply choose top k frequent tags and train k composition functions. A common composition function is shared across phrases with all infrequent tags. The value of k depends on the size of the training set and the occurrences of each tag. Specially, when $k = 0$, the model is the same as the traditional RNN.

4.2 Tag Embedded RNN and RNTN (TE-RNN/RNTN)

In this section, we propose tag embedded RNN (TE-RNN) and tag embedded RNTN (TE-RNTN) to respect the part-of-speech tags of child nodes during composition. As mentioned above, tags of parent nodes have impact on composition. However, some phrases with the same tag should be composed in different ways. For example, *'is interesting'* and *'like swimming'* have the same tag *VP*. But it is not reasonable to compose the two phrases using the previous model because the part-of-speech tags of their children are quite different. If we use different composition functions for children with different tags like TG-RNN, the number of tag pairs will amount to as many as $k \times k$, which makes the models infeasible due to too many parameters.

In order to capture the compositional effects of the tags of child nodes, an embedding $e_t \in R^{d_e}$ is created for every tag t, where d_e is the dimension of tag vector. The tag vector and phrase vector are

concatenated during composition as illustrated in Figure 4.

Formally, the phrase vector is composed by the function

$$g(v_i^l, e_{t_i^l}, v_i^r, e_{t_i^r}) = W \begin{bmatrix} v_i^l \\ e_{t_i^l} \\ v_i^r \\ e_{t_i^r} \end{bmatrix} + b \qquad (6)$$

where t_i^l and t_i^r are tags of the left and the right nodes respectively, $e_{t_i^l}$ and $e_{t_i^r}$ are tag vectors, and $W \in R^{d\times(2d_e+2d)}$ is the composition matrix. We term this model Tag embedded RNN, TE-RNN for short.

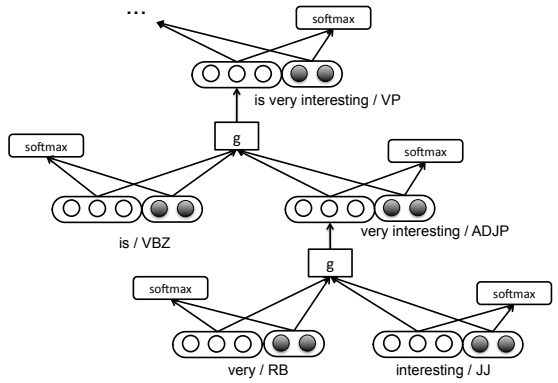

Figure 4: RNN with tag embedding. There is a tag embedding table, storing vectors for *RB*, *JJ*, and *ADJP*, etc. Then we compose the phrase vector *'very interesting'* from the vectors for *'very'* and *'interesting'*, and the tag vectors for *RB* and *JJ*.

Similarly, this idea can be applied to Recursive Neural Tensor Network (Socher et al., 2013b). In RNTN, the tag vector and the phrase vector can be interweaved together through a tensor. More specifically, the phrase vectors and tag vectors are multiplied by the composed tensor. The composition function changes to the following:

$$\begin{aligned} & g(v_i^l, e_{t_i^l}, v_i^r, e_{t_i^r}) \\ = & \begin{bmatrix} v_i^l \\ e_{t_i^l} \\ v_i^r \\ e_{t_i^r} \end{bmatrix} T^{[1:d]} \begin{bmatrix} v_i^l \\ e_{t_i^l} \\ v_i^r \\ e_{t_i^r} \end{bmatrix} + W \begin{bmatrix} v_i^l \\ e_{t_i^l} \\ v_i^r \\ e_{t_i^r} \end{bmatrix} + b \end{aligned} \qquad (7)$$

where the variables are similar to those defined in equation 3 and equation 7. We term this model Tag embedded RNTN, TE-RNTN for short.

The phrase vectors and tag vectors are used as input to a softmax classifier, giving the posterior probability over labels via

$$y_i = \text{softmax}(W_s \begin{bmatrix} v_i \\ e_{t_i} \end{bmatrix} + b_s) \qquad (8)$$

4.3 Model Training

Let y_i be the target distribution for node i, $\hat{y_i}$ be the predicted sentiment distribution. Our goal is to minimize the cross-entropy error between y_i and $\hat{y_i}$ for all nodes. The loss function is defined as follows:

$$E(\theta) = -\sum_i \sum_j y_i^j \log \hat{y_i}^j + \lambda ||\theta||^2 \qquad (9)$$

where j is the label index, λ is a L_2-regularization term, and θ is the parameter set.

Similar to RNN, the parameters for our models include word vector table L, the composition matrix W, and the sentiment classification matrix W_s. Besides, our models have some additional parameters, as discussed below:

TG-RNN: There are k composition matrices for top k frequent tags. They are defined as $W_t \in R^{k\times d\times 2d}$. The original composition matrix W is for all infrequent tags. As a result, the parameter set of TG-RNN is $\theta = (L, W, W_t, W_s)$.

TE-RNN: The parameters include the tag embedding table E, which contains all the embeddings for part-of-speech tags for words and phrases. And the size of matrix $W \in R^{d\times(2d+2d_e)}$ and the softmax classifier $W_s \in R^{N\times(d_e+d)}$. The parameter set of TE-RNN is $\theta = (L, E, W, W_s)$.

TE-RNTN: This model has one more tensor $T \in R^{(2d+2d_e)\times(2d+2d_e)\times d}$ than TE-RNN. The parameter set of TE-RNTN is $\theta = (L, E, W, T, W_s)$

5 Experiment

5.1 Dataset and Experiment Setting

We evaluate our models on Stanford Sentiment Treebank which contains fully labeled parse trees. It is built upon 10,662 reviews and each sentence has sentiment labels on each node in the parse tree. The sentiment label set is {0,1,2,3,4}, where the numbers mean *very negative*, *negative*, *neutral*, *positive*, and *very positive*, respectively. We use standard split (train: 8,544 dev: 1,101, test: 2,210) on the corpus in our experiments. In addition, we add the part-of-speech tag for each leaf node and phrase-type tag for each interior node

using the latest version of Stanford Parser. Because the newer parser generated trees different from those provided in the datasets, 74/11/11 reviews in train/dev/test datasets are ignored. After removing the broken reviews, our dataset contains 10566 reviews (train: 8,470, dev: 1,090, test: 2,199).

The word vectors were pre-trained on an unlabeled corpus (about 100,000 movie reviews) by word2vec (Mikolov et al., 2013b) as initial values and the other vectors is initialized by sampling from a uniform distribution $\mathcal{U}(-\epsilon, \epsilon)$ where ϵ is 0.01 in our experiments. The dimension of word vectors is 25 for RNN models and 20 for RNTN models. *Tanh* is chosen as the nonlinearity function. And after computing the output of node i with $v_i = f(g(v_i^l, v_i^r))$, we set $v_i = \frac{v_i}{||v_i||}$ so that the resulting vector has a limited norm. Backpropagation algorithm (Rumelhart et al., 1986) is used to compute gradients and we use minibatch SGD with momentum as the optimization method, implemented with Theano (Bastien et al., 2012). We trained all our models using stochastic gradient descent with a batch size of 30 examples, momentum of 0.9, L_2-regularization weight of 0.0001 and a constant learning rate of 0.005.

5.2 System Comparison

We compare our models with several methods which are evaluated on the Sentiment Treebank corpus. The baseline results are reported in (Dong et al., 2014) and (Kim, 2014).

We make comparison to the following baselines:

- **SVM.** A SVM model with bag-of-words representation (Pang and Lee, 2008).
- **MNB/bi-MNB.** Multinomial Naive Bayes and its bigram variant, adopted from (Wang and Manning, 2012).
- **RNN.** The first Recursive Neural Network model proposed by (Socher et al., 2011).
- **MV-RNN.** Matrix Vector Recursive Neural Network (Socher et al., 2012) represents each word and phrase with a vector and a matrix. As reported, this model suffers from too many parameters.
- **RNTN.** Recursive Neural Tenser Network (Socher et al., 2013b) employs a tensor for composition function which could model the meaning of longer phrases and capture negation rules.
- **AdaMC.** Adaptive Multi-Compositionality for RNN and RNTN (Dong et al., 2014) trains more than one composition functions and adaptively learns the weight for each function.
- **DCNN/CNN.** Dynamic Convolutional Neural Network (Kalchbrenner et al., 2014) and a simple Convolutional Neural Network (Kim, 2014), though these models are of different genres to RNN, we include them here for fair comparison since they are among top performing approaches on this task.
- **Para-Vec.** A word2vec variant (Le and Mikolov, 2014) that encodes paragraph information into word embedding learning. A simple but very competitive model.
- **DRNN.** Deep Recursive Neural Network (Irsoy and Cardie, 2014) stacks multiple recursive layers.

Method	Fine-grained	Pos./Neg.
SVM	40.7	79.4
MNB	41.0	81.8
bi-MNB	41.9	83.1
RNN	43.2	82.4
MV-RNN	44.4	82.9
RNTN	45.7	85.4
AdaMC-RNN	45.8	87.1
AdaMC-RNTN	46.7	88.5
DRNN	49.8	87.7
TG-RNN (ours)	47.0	86.3
TE-RNN (ours)	48.0	86.8
TE-RNTN (ours)	48.9	87.7
CNN	48.0	88.1
DCNN	48.5	86.8
Para-Vec	48.7	87.8

Table 1: Classification accuray. *Fine-grained* stands for 5-class prediction and *Pos./Neg.* means binary prediction which ignores all neutral instances. All the accuracy is at the sentence level (root).

The comparative results are shown in Table 1. As illustrated, **TG-RNN** outperforms RNN, RNTN, MV-RNN, AdMC-RNN/RNTN.

Compared with RNN, the fine-grained accuracy and binary accuracy of TG-RNN is improved by 3.8% and 3.9% respectively. When compared with AdaMC-RNN, the accuracy of our method rises by 1.2% on the fine-grained prediction. The results show that the syntactic knowledge does facilitate phrase vector composition in this task.

As for **TE-RNN/RNTN**, the fine-grained accuracy of TE-RNN is boosted by 4.8% compared with RNN and the accuracy of TE-RNTN by 3.2% compared with RNTN. TE-RNTN also beat the AdaMC-RNTN by 2.2% on the fine-grained classification task. TE-RNN is comparable to CNN and DCNN, another line of models for this task. TE-RNTN is better than CNN, DCNN, and ParaVec, which are the top performing approaches on this task. TE-RNTN is worse than DRNN, but the complexity of DRNN is much higher than TE-RNTN, which will be discussed in the next section. Furthermore, TE-RNN is also better than TG-RNN. This implies that learning the tag embeddings for child nodes is more effective than simply using the tags of parent phrases in composition.

Note that the fine-grained accuracy is more convincible and reliable to compare different approaches due to the two facts: First, for the binary classification task, some approaches train another binary classifier for positive/negative classification while other approaches, like ours, directly use the fine-grained classifier for this purpose. Second, how the neutral instances are processed is quite tricky and the details are not reported in the literature. In our work, we simply remove neural instances from the test data before the evaluation. Let the 5-dimension vector y be the probabilities for each sentiment label in a test instance. The prediction will be positive if $\arg\max_{i,i\neq 2} y_i$ is greater than 2, otherwise negative, where $i \in \{0, 1, 2, 3, 4\}$ means very negative, negative, neutral, positive, very positive, respectively.

5.3 Complexity Analysis

To gain deeper understanding of the models presented in Table 1, we discuss here about the parameter scale of the RNN/RNTN models since the prediction power of neural network models is highly correlated with the number of parameters.

The analysis is presented in Table 2 (the optimal values are adopted from the cited papers). The parameters for the word table have the same size $n \times d$ across all recursive neural models, where n is the number of words and d is the dimension of word vector. Therefore, we ignore this part but focus on the parameters of composition functions, termed *model size*. Our models, TG-RNN/TE-RNN, have much less parameters than RNTN and AdMC-RNN/RNTN, but have much better performance. Although TE-RNTN is worse than DRNN, however, the parameters of DRNN are almost 9 times of ours. This indicates that DRNN is much more complex, which requires much more data and time to train. As a matter of a fact, our TE-RNTN only takes 20 epochs for training which is 10 times less than DRNN.

Method	model size	# of parameters
RNN	$2d^2$	$1.8K$
RNTN	$4d^3$	$108K$
AdaMC-RNN	$2d^2 \times c$	$18.7K$
AdaMC-RNTN	$4d^3 \times c$	$202K$
DRNN	$d \times h \times l$ $+2h^2 \times l$	$451K$
TG-RNN (ours)	$2d^2 \times (k+1)$	$8.8K$
TE-RNN (ours)	$2(d+d_e) \times d$	$1.7K$
TE-RNTN (ours)	$4(d+d_e)^2 \times d$	$54K$

Table 2: The model size. d is the dimension of word/phrase vectors (the optimal value is 30 for RNN & RNTN, 25 for AdaMC-RNN, 15 for AdaMC-RNTN, 300 for DRNN). For AdaMC, c is the number of composition functions (15 is the optimal setting). For DRNN, l and h is the number of layers and the width for each layer (the optimal values $l = 4$, $h = 174$). For our methods, k is the number of unshared composition matrices and d_e the dimension of tag embedding, for the optimal setting refer to Section 5.4.

5.4 Parameter Analysis

We have two key parameters to tune in our proposed models. For **TG-RNN**, the number of composition functions k is an important parameter, which corresponds to the number of distinct POS tags of phrases.

Let's start from the corpus analysis. As shown in Table 3, the corpus contains 215,154 phrases but the distribution of phrase tags is extremely imbalanced. For example, the phrase tag *'NP'* appears 60,239 times while *'NAC'* appears only 10 times. Hence, it is impossible to learn a composi-

Phrase tag	Frequency	Phrase tag	Frequency
NP	60,239	ADVP	1,140
S	33,138	PRN	976
VP	26,956	FARG	792
PP	14,979	UCP	362
ADJP	7,912	SSINV	266
SBAR	5,308	others	1,102

Table 3: The distribution of phrase-type tags in the training data. The top 6 frequency tags cover more than 95% phrases.

tion function for the infrequent phrase tags.

Each of the top k frequent phrase tags corresponds to a unique composition function, while all the other phrase tags share a same function. We compare different k for TG-RNN. The accuracy is shown in Figure 5. Our model obtains the best performance when k is 6, which is accordant with the statistics in Table 3.

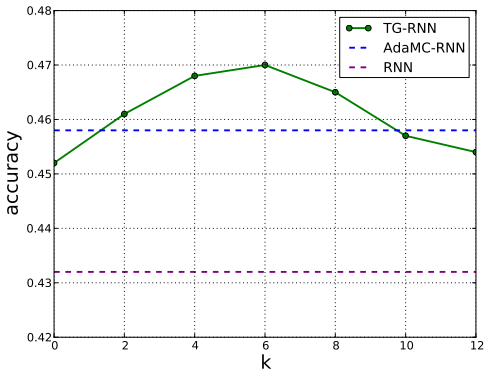

Figure 5: The accuracy for TG-RNN with different k.

For **TE-RNN/RNTN**, the key parameter to tune is the dimension of tag vectors. In the corpus, we have 70 types of tags for leaf nodes (words) and interior nodes (phrases). Infrequent tags whose frequency is less than 1,000 are ignored. There are 30 tags left and we learn an embedding for each of these frequent tags. We varies the dimension of the embedding d_e from 0 to 30.

Figure 6 shows the accuracy for TE-RNN and TE-RNTN with different dimensions of d_e. Our model obtains the best performance when d_e is 8 for TE-RNN and 6 for TE-RNTN. The results show that too small dimensions may not be sufficient to encode the syntactic information of tags and too large dimensions damage the performance.

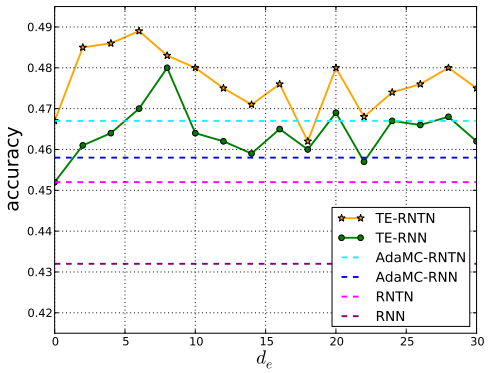

Figure 6: The accuracy for TE-RNN and TE-RNTN with different dimensions of d_e.

5.5 Tag Vectors Analysis

In order to prove tag vectors obtained from tag embedded models are meaningful, we inspect the similarity between vectors of tags. For each tag vector, we find the nearest neighbors based on Euclidean distance, summarized in Table 4.

Tag	Most Similar Tags
JJ (Adjective)	ADJP (Adjective Phrase)
VP (Verb Phrase)	VBD (past tense) VBN (past participle)
. (Dot)	: (Colon)

Table 4: Top 1 or 2 nearest neighboring tags with definition in brackets.

Adjectives and verbs are of significant importance in sentiment analysis. Although '*JJ*' and '*ADJP*' are word and phrase tag respectively, they have similar tag vectors, because of playing the same role of *Adjective* in sentences. '*VP*', '*VBD*' and '*VBN*' with similar representations all represent verbs. What is more interesting is that the nearest neighbor of dot is colon, probably because both of them are punctuation marks. Note that tag classification is none of our training objectives and surprisingly the vectors of similiar tags are clustered together, which can provides additional information during sentence composition.

6 Conclusion

In this paper, we present two ways to leverage syntactic knowledge in Recursive Neural Networks.

The first way is to use different composition functions for phrases with different tags so that the composition processing is guided by phrase types (TG-RNN). The second way is to learn tag embeddings and combine tag and word embeddings during composition (TE-RNN/RNTN). The proposed models are not only effective (w.r.t competing performance) but also efficient (w.r.t well-controlled parameter scale). Experiment results show that our models are among the top performing approaches up to date, but with much less parameters and complexity.

Acknowledgments

This work was partly supported by the National Basic Research Program (973 Program) under grant No.2012CB316301/2013CB329403, the National Science Foundation of China under grant No.61272227/61332007, and the Beijing Higher Education Young Elite Teacher Project. The work was also supported by Tsinghua University Beijing Samsung Telecom R&D Center Joint Laboratory for Intelligent Media Computing.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

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

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*. AAAI.

Karl Moritz Hermann and Phil Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *ACL*, pages 894–904. Association for Computer Linguistics.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *NIPS*, pages 2096–2104.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665. Association for Computer Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751. Association for Computational Linguistics.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, volume 32, pages 1188–1196.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for ITG-based translation. In *EMNLP*, pages 567–577. Association for Computer Linguistics.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Sebastian Padó and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *ACL*, pages 907–916. Association for Computer Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1):159–216.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*, pages 1201–1211. Association for Computational Linguistics.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013a. Parsing with compositional vector grammars. In *ACL*, pages 455–465. Association for Computer Linguistics.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642. Association for Computational Linguistics.

Peter D Turney, Patrick Pantel, et al. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188.

Sida I Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*, pages 90–94. Association for Computational Linguistics.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *EMNLP*, pages 172–182. Association for Computer Linguistics.