The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Training and Evaluating Improved Dependency-Based Word Embeddings

Chen Li,[†] Jianxin Li,[†] Yangqiu Song,[‡] Ziwei Lin[†]
[†]Department of Computer Science & Engineering, Beihang University, Beijing 100191, China
[‡]Department of Computer Science & Engineering, Hong Kong University of Science and Technology, Hong Kong
{lichen,lijx,linzw}@act.buaa.edu.cn, yqsong@cse.ust.hk

Abstract

Word embedding has been widely used in many natural language processing tasks. In this paper, we focus on learning word embeddings through selective higher-order relationships in sentences to improve the embeddings to be less sensitive to local context and more accurate in capturing semantic compositionality. We present a novel multi-order dependency-based strategy to composite and represent the context under several essential constraints. In order to realize selective learning from the word contexts, we automatically assign the strengths of different dependencies between co-occurred words in the stochastic gradient descent process. We evaluate and analyze our proposed approach using several direct and indirect tasks for word embeddings. Experimental results demonstrate that our embeddings are competitive to or better than state-of-the-art methods and significantly outperform other methods in terms of context stability. The output weights and representations of dependencies obtained in our embedding model conform to most of the linguistic characteristics and are valuable for many downstream tasks.

Introduction

Distributed word representations, also known as word embeddings, have attracted more attention and been widely applied to many Natural Language Processing (NLP) tasks (Mikolov et al. 2013b; Collobert et al. 2011; Jeffrey, Richard, and Christopher 2014; Mikolov et al. 2013a; Turian, Ratinov, and Bengio 2010; Levy, Goldberg, and Dagan 2015). Based on the distributed hypothesis, such models fit the co-occurrence of words using vector representations. The output vectors can be use not only to explore relationships between words (Mikolov et al. 2013b; 2013a), but also to achieve remarkable effects in many NLP tasks, such as Part-Of-Speech (POS) tagging and Named Entity Recognition (NER) (Bengio, Courville, and Vincent 2013; Baroni, Dinu, and Kruszewski 2014; Collobert et al. 2011). Most of the embedding models iteratively learn the characteristics of co-occurrence, though the structures of models can be very different. For example, some models predict target word by considering its local context, such as C&W (Collobert et al. 2011) and Continuous Bag-Of-Words (CBOW) (Mikolov et al. 2013a), while the other methods use target word to forecast its surrounding words, such as Skip-Gram (SG) (Mikolov et al. 2013b) and its variants (Qiu et al. 2014).

S1: Dogs are **usually** highly variable in height and weight.
S2: **Usually**, domestic cats are similar in size.

	BOW	First-Order	Second-Order
S1	Dogs are highly variable	*advmod*$^{-1}$ /variable	*nmod-advmod*$^{-1}$/height *nsubj-advmod*$^{-1}$/Dogs *cop-advmod*$^{-1}$/are *advmod-advmod*$^{-1}$/highly
S2	domestic cats	*advmod*$^{-1}$ /similar	*nmod-advmod*$^{-1}$/size *nsubj-advmod*$^{-1}$/cats *cop-advmod*$^{-1}$/are

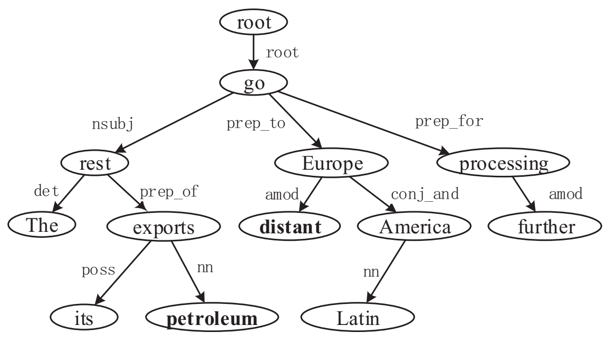
Table 1: First- and second-order dependency-based context of two similar sentences (S1 and S2) for the same word "usually." First-order dependency-based context comes from one-hop neighborhoods in the dependency parse tree where second-order context comes from two-hop neighborhoods.

The above intuitions to train word embeddings have been proven to be useful, however, some obvious issues have been found in the composition and representation of local context. First, though the large-scale corpus improves the learning effect of embeddings, with more diverse of a word's context may result in more difficulty to use one single embedding vector to approximate the co-occurrences. Using multiple embedding vectors of a word may be able to solve this problem (Huang et al. 2012), but how to decide the number of vectors of a word is much more difficult.

Second, in most of the existing models, context words are treated equally or distinguished by distances between the context words and the target central word. However, this does not conform to the real linguistic rules and human cognition (Levy and Goldberg 2014; Komninos and Manandhar 2016). As illustrated in Table 1, the word "cats" is more significant than word "usually" for target word "domestic," because the dependency *amod* (between "cats" and "domestic") is more important than other dependencies.

In this paper, we generalize word embedding approaches to context combining context-dependent weights and multi-order dependencies. Our approach has two major contributions:

1) We propose a fast and efficient strategy to generate the



word	Second-order Dependency	Multi-order Dependency (sorted by $score_w$)
distant	*amod*$^{-1}$/Europe *prep_to*$^{-1}$-*amod*$^{-1}$/go *conj_and-amod*$^{-1}$/America	0.9384/Europe 0.9276/America 0.7719/Lation 0.4732/exports ...
petroleum	*nn*$^{-1}$/exports *poss-nn*$^{-1}$/its *prep_of*$^{-1}$-*nn*$^{-1}$/rest	1.1106/exports 0.9338/processing 0.9173/rest 0.5532/further ...

(a) The Syntactic Dependency Tree (SDT) of the sentence. (b) Different dependency-based context composition.

Figure 1: An instance for the composition of local context. Sentence: The rest of its **petroleum** exports go to **distant** Europe and Latin America for further processing.

context based on the Syntactic Dependency Parse (SDP), which not only composites the local context without losing any important information but also ensures the stability of semantic meaning of word embeddings when there are much diverse context with large training corpus.

2) We determine the final representation of context by appending dependency-based parameters into our models. The modified model updates dependency parameters in real-time to achieve adaptive determination of the importance of the context.

To evaluate the effectiveness of our method, we develop our algorithm based on the Word2Vec tool, which is simple, efficient and has comparable performance to other word embedding models (Mikolov et al. 2013b; 2013a). We can achieve competitive experimental results in different NLP tasks. We show that our integrated model can extract more linguistic features from large-scale corpus through case studies. We also conducted detailed analysis of the dependency parameters with respect to several interesting phenomena. Our system is publicly available at https://github.com/RingBDStack/dependency-based-w2v.

Related Work

In this section, we briefly review the related work. As a mature grammar applied in many NLP tasks, SDP has been also used to composite local context in recent distributed representation learning models. Levy and Goldberg (2014) introduced SDP into word embedding by presenting a simple arbitrary word context and obtaining specific word embeddings with abundant structural and functional features. Qiu, Zhang, and Lu (2015) emphasized linguistic features in a sentence, and confirmed the effect of SDP through analyzing some representative experiments with dependency-based word embeddings. Besides, Komninos and Manandhar (2016) introduced the second-order dependency into the context composition in word embeddings and verified its ability to capture features by several tasks, such as sentence classification, analogy, and so on. In particular, by using a Recurrent Neural Network (RNN), Yin et al. (2016) trained representations of multi-order dependency sequences between a target word and its context words considering the spatial distance between them. The embeddings can significantly improve the CRF based aspect term extraction[1]. In addition, the embedding with SDP based contexts can also be used to improve neural machine translation (Eriguchi, Hashimoto, and Tsuruoka 2016; Chen et al. 2017). There have been also other dependency-based embeddings to improve other NLP tasks, such as NER, natural language understanding, and question answering (Jie, Muis, and Lu 2017; Roy and Roth 2017; Xiang et al. 2016). However, all the existing approaches empirically set the number of dependency order of the composition context and the selected context words are treated equally.

Improved Dependency-Based Word Embeddings

In this section, we first briefly review the existing methods for compositing context and then introduce our proposed method.

Existing Methods for Compositing Context

Original word embedding uses local window to construct a word's context (Collobert et al. 2011; Mikolov et al. 2013a). SDP has been proposed to capture the longer-distance dependencies that are syntactically related to the target words (Levy and Goldberg 2014; Komninos and Manandhar 2016; Qiu, Zhang, and Lu 2015; Yin et al. 2016). However, all the existing methods are based on a simple hypothesis that the relation-strength between two words is inversely proportional to the order of dependency between them, which may not hold in certain circumstances. For example, in Figure 1(b), choosing all the first-order and second-order dependent words in SDT will lose some important information and bring lots of redundant or noisy in-

[1]Because we use general local linear method to composite its context and the objective used by Yin et al. (2016) is different from our method, we do not choose it for a comparison.

Number	1	2	3	4	5	6	7	8	9	≥ 10
Percentage	7.91%	8.64%	9.71%	11.56%	13.1%	13.47%	11.92%	10.13%	7.84%	$\approx$ 5.72%

Table 2: The distribution of the number of first-order dependent words in dependency parse tree of all target words in 3.7 million sentences from English Wikipedia (there are about fifteen words in each sentence on average). For example, The words *rest*, *Europe* and *exports* in Figure 1 have 3 first-order dependencies, but *Latin* has only 1 first-order dependency.

formation into the context. More severely, as shown in Table 2, the numbers of first-order dependency of each single word distribute evenly, which means the fixed low-order dependency window will cause partial target words to have too few or too many words in their context (e.g., only a single word or the whole sentence as the dependency-based context). Second-order dependency-based context also has the same problem. Hence, many words (nearly 30% $\sim$ 40%) in the vocabulary can not get a corresponding high-quality composite context with stable linguistic features.

Multi-Order Dependency-Based Composite Context

Instead of simply combining the words with different dependency order, we propose an adaptive approach to composite the context.

Alternative Multi-Order Dependency Set As illustrated in Figure 1(a), all of the syntactic dependencies among different words have been defined in SDT (we regard the inverse of a relation as a different dependency, e.g. *amod* and *amod*$^{-1}$ are treated differently). Some common prepositions are included as special syntactic dependencies as well (e.g. *by, to, of*). Then, we derive the multi-order dependency set as follows: for a target word w with its multi-order dependency related words w_1,w_2,...,w_k, we consider all of the units in context d_{w,w_i} $i \leq k$, where d_{w,w_i} is the type of the multi-order SDP between target word w and its dependency related word w_i (e.g. *amod, prep_to, amod-conj_and*$^{-1}$), as the alternative multi-order dependency set $\mathcal{D}(w)$.

Composite Principle In order to reserve the most valuable words and to provide corresponding weight for each word during the training process, the words in alternative multi-order dependency set $\mathcal{D}(w)$ need to be scored based on a comprehensive consideration of the different dependencies and order between context word and target word. We denote $w_{i,1}, \ldots, w_{i,n}$ to be the path from $w = w_{i,1}$ to $w_i = w_{i,n}$ if w_i is a nth order dependency word of w. Then the score of w_i related to w is defined as:

$$s_{d_{w,w_i}} = \prod_{j=2}^{n} \frac{\varphi_{d_{w_{i,j-1},w_{i,j}}}}{\lambda_j}, \tag{1}$$

where $\varphi_{d_{w_{i,j-1},w_{i,j}}}$ denotes the weight of dependency between w_{j-1} and w_j and λ_j is the penalty coefficient for jth-order, which is preset before training. The final word set $\mathcal{C}_{\mathcal{D}}(w)$ can be constituted by sorting all of the elements in $\mathcal{D}(w)$ according to their scores directly (as shown in Figure 1(b)).

Training Dependency-Based Word Embeddings

In this section, we introduce the CBOW based on Hierarchical Softmax (HS) and SG with Negative Sampling (NS), and modify them to realize our proposed dependency-based composite context.

Dependency-Based CBOW&HS The original CBOW model sums $2n$ words surrounding target word w as its composite context $\mathcal{C}(w)$, and maximizes the log-likelihood function $\sum_w \log p(w|\mathcal{C}(w))$ as the training objective function. To integrate the dependency information into this model, $s_{d_{w,w_i}}$ and dependency representation $v(d_{w,w_i})$ are introduced into training process. More specifically, the dependency-based (generalized) CBOW model concatenates the dependency representation $v(d_{w,w_i})$ with the word vector $v(w_i)$ as a new input vector $v'(w_i)$ (as shown in Figure 2). Then, all of the $v'(w_i)$ are aggregated as new dependency-based context representation $x_w = \sum_{i=1,w_i \in \mathcal{C}_{\mathcal{D}}(w)}^{c_w} s_{d_{w,w_i}} \cdot v'(w_i)/c_w$, where c_w is the size of context window.

Hence, the dependency-based CBOW, drawing on HS, redefines the probability $p(w|\mathcal{C}_{\mathcal{D}}(w))$ by calculating from the root node to the leaf node (target word w) in Huffman tree. Then, the object function of model can be re-written as:

$$\mathcal{L}_{CBOW\&HS}^{Deps} = \sum_w \log p(w|\mathcal{C}_{\mathcal{D}}(w)) = \sum_w \sum_{i=2}^{l^w} \mathcal{L}(w,i), \tag{2}$$

where l^w is the length of the path from the root node to the leaf node, $\mathcal{L}(w,i) = (1-d_i^w)\cdot\log[\sigma(x_w^\top \theta_{i-1}^w)] + d_i^w \cdot \log[1-\sigma(x_w^\top \theta_{i-1}^w)]$ is the abbreviated form for the classified result on i-th non-leaf node as the $d_i^w \in \{0,1\}$, θ_{i-1}^w denotes the parameter vector of i-th node, and $\sigma(x) = 1/(1+\exp(-x))$ is the activation function.

Via the stochastic gradient based method, the parameters θ_{i-1}^w and the original words $\widetilde{w} \in \mathcal{C}_{\mathcal{D}}(w)$ can be updated as shown in what follows:

$$\theta_{i-1}^w := \theta_{i-1}^w + \eta[1 - d_i^w - \sigma(x_w^\top \theta_{i-1}^w)]x_w \tag{3}$$

and

$$v(\widetilde{w}) := v(\widetilde{w}) + \eta f(\sum_{i=2}^{l^w} \frac{\partial \mathcal{L}(w,i)}{\partial x_w}), \tag{4}$$

where $\eta \in (0,1)$ is the learning rate of the whole model, which is inversely proportional to the number of iterations (Hegde, Indyk, and Schmidt 2015), and $f(\cdot)$ is the function to intercept the part of $v(\widetilde{w})$'s gradient.

We also update the dependency weight $\varphi_{d(w,w_i)}$ and dependency representation $v(d_{w,w_i})$ during the training process. Those parameters can effectively and diacritically identify the most useful context for target word in sentences

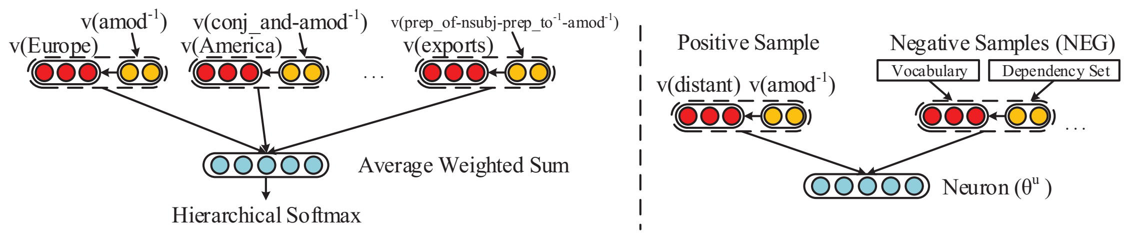


Figure 2: The overview of proposed modified models (take *distant* in Table 1 as example). Left: Dependency-based CBOW&HS. Right: Dependency-based SG&NS.

without considering any limits and supervisions (such as the order or the distance, etc.). In dependency-based CBOW & HS, we can obtain partial derivative of $s_{d_{w,w_j}}$ as follow:

$$\begin{aligned}\frac{\partial \mathcal{L}(w,i)}{\partial s_{d_{w,w_j}}} &= \frac{\partial \mathcal{L}(w,i)}{\partial x_w} \cdot \frac{\partial x_w}{\partial s_{d_{w,w_j}}} \\ &= [1 - d_i^w - \sigma(x_w^\top \theta_{i-1}^w)]\theta_{i-1}^w \cdot \frac{v'(w_j)}{c_w}.\end{aligned} \tag{5}$$

Then, the dependency parameters $\varphi_{d_{w_{i,k-1},w_{i,k}}}$ and $v(d_{w_{i,k-1},w_{i,k}})$ can be updated as shown:

$$\varphi_{d_{w_{i,k-1},w_{i,k}}} := \varphi_{d_{w_{i,k-1},w_{i,k}}} + \eta' \sum_{i=2}^{l^w} \frac{\partial \mathcal{L}(w,i)}{\partial s_{d_{w,w_j}}} \tag{6}$$

and

$$v(d_{w_{i,k-1},w_{i,k}}) := v(d_{w_{i,k-1},w_{i,k}}) + \eta' f(\sum_{i=2}^{l^w} \frac{\partial \mathcal{L}(w,i)}{\partial x_w}), \tag{7}$$

where $\eta' = \eta / \prod_{w_j \in \mathcal{C}_\mathcal{D}(w), l=1}^{k} \lambda_l$, c_w is the size of context window, $f(\cdot)$ is the function to intercept the part of dependency representation's gradient and i, j, k respectively denotes the ith non-leaf node, jth word in $\mathcal{C}_\mathcal{D}(w)$, kth-order dependency between target word and its context word.

Dependency-based SG&NS In contrast to the CBOW, the original SG model uses the target word to predict its contexts. Thus, its training object is to maximize the log-likelihood function $\sum_w \log p(\mathcal{C}(w)|w) = \sum_w \sum_{u \in \mathcal{C}(w)} \log p(u|w)$.

As shown in Figure 2, aiming to add the dependency information, we randomly select negative examples from the whole vocabulary and dependency set according to their frequencies, and concatenate them together as negative examples $v'(\widetilde{w})$ in set $NEG(w)$. Moreover, $x_{\widetilde{w}} = s_{\widetilde{w}} \cdot v'(\widetilde{w})$ is treated as the final representation of negative example.

Hence, the dependency-based SG, drawing on NS, takes $g(w) = \prod_{\widetilde{w} \in \mathcal{C}_\mathcal{D}(w)} \prod_{u \in \{w\} \cup NEG^{\widetilde{w}}(w)} p(u|\widetilde{w})$ as the objective function, which can be redefined as follows:

$$\mathcal{L}_{SG\&NS}^{Deps} = \sum_w \log g(w) = \sum_w \sum_{\widetilde{w} \in \mathcal{C}_\mathcal{D}(w)} \sum_{u \in \{w\} \cup NEG^{\widetilde{w}}(w)} \mathcal{L}(w, \widetilde{w}, u), \tag{8}$$

where $\mathcal{L}(w, \widetilde{w}, u) = L^w(u) \cdot \log[\sigma(x_{\widetilde{w}}^\top \theta^u)] + [1 - L^w(u)] \cdot \log[1 - \sigma(x_{\widetilde{w}}^\top \theta^u)]$, $\sigma(x)$ has the same meaning in Equation (3), θ^u denotes the parameter vector of NS neuron, and $L^w(u) \in \{0, 1\}$, which depends that the u is a positive example or negative example. Then, the updating functions are also as follows:

$$\theta^u := \theta^u + \eta[L^w(u) - \sigma(x_{\widetilde{w}}^\top \theta^u)] x_{\widetilde{w}} \tag{9}$$

and

$$v(\widetilde{w}) := v(\widetilde{w}) + \eta f(\sum_{u \in \{w\} \cup NEG(w)} \frac{\partial \mathcal{L}(w, \widetilde{w}, u)}{\partial x_{\widetilde{w}}}), \tag{10}$$

where $f(\cdot)$ is the function to intercept the part of $v(\widetilde{w})$'s gradient.

In analogy to the previous section, we know that the partial derivative of s_u are obtained with the same method, which is shown as follow:

$$\begin{aligned}\frac{\partial \mathcal{L}(w, \widetilde{w}, u)}{\partial s_u} &= \frac{\partial \mathcal{L}(w, \widetilde{w}, u)}{\partial x_{\widetilde{w}}} \cdot \frac{\partial x_{\widetilde{w}}}{\partial s_u} \\ &= [L^w(u) - \sigma(x_{\widetilde{w}}^\top \theta^u)]\theta^u \cdot v'(\widetilde{w}).\end{aligned} \tag{11}$$

Thus, the dependency parameters $\varphi_{d_{w_{i,k-1},w_{i,k}}}$ and $v(d_{w_{i,k-1},w_{i,k}})$ can be updated as shown:

$$\varphi_{d_{w_{i,k-1},w_{i,k}}} := \varphi_{d_{w_{i,k-1},w_{i,k}}} + \eta' \sum_{i=2}^{l^w} \frac{\partial \mathcal{L}(w, \widetilde{w}, u)}{\partial s_u} \tag{12}$$

and

$$v(d_{w_{i,k-1},w_{i,k}}) := v(d_{w_{i,k-1},w_{i,k}}) + \eta' f(\sum_{i=2}^{l^w} \frac{\partial \mathcal{L}(w, \widetilde{w}, u)}{\partial x_{\widetilde{w}}}), \tag{13}$$

where $\eta' = \eta / \prod_{w_j \in \mathcal{C}_\mathcal{D}(w), l=1}^{k} \lambda_l$, c_w is the size of context window, $f(\cdot)$ is the function to intercept the part of dependency representation's gradient and k denotes the kth-order dependency between target word and its context word.

Optimization

In order to improve the overall efficiency and effectiveness of the modified models, we introduce some optimization tricks here. The redundant information will be increased and the performance and output effect will be reduced indirectly, if the size of context window is too large. Considering comprehensively the properties of large-scale corpus (imbalance,

	SIMLEX-999	TR-3K	WS-353-SIM	VERB-143	MC-30	MTurk-287	RG-65	RW	YP-130
GloVe	0.2627	0.6092	0.5670	0.3171	0.4653	0.6145	0.5458	0.2641	0.4016
CBOW&HS	0.3197	0.6694	0.7068	0.3706	0.6843	0.6486	0.6789	0.3576	0.3203
SG&NS	0.3559	0.6963	0.7361	0.4041	0.7380	0.6532	0.7039	0.3730	0.4323
Deps	**0.3705**	0.6742	0.6899	0.3844	0.7426	0.6371	0.6713	**0.3994**	0.4446
EXT	0.3563	0.6814	0.7058	0.3879	0.7391	0.6406	0.6711	0.3724	0.4358
CBOW$_{Deps}$&HS (only weight)	0.3202	0.6732	0.7094	0.3824	0.6883	0.6533	0.6809	0.3607	0.3215
SG$_{Deps}$&NS (only weight)	0.3613	0.7029	0.7393	0.4084	0.7488	0.6594	0.7267	0.3894	0.4483
CBOW$_{Deps}$&HS	0.3291	0.6775	0.7234	0.3942	0.7042	0.6524	0.6875	0.3619	0.3311
SG$_{Deps}$&NS	**0.3686**	**0.7109**	**0.7456**	**0.4122**	**0.7552**	**0.6647**	**0.7348**	**0.3964**	**0.4533**

Table 3: Comparison of word embeddings for word similarity/relatedness (the average of several trials).

huge and complex), we dynamically adjust the context window size of target word w as follows:

$$c_w = \max\left(size_{max} - \log f_w, size_{min}\right), \tag{14}$$

where f_w is the frequency of word w in corpus, $size_{max}$ and $size_{min}$ are artificial parameters for restricting the size of window.

Meanwhile, since the distribution of dependencies, similar to that of word frequency, is greatly uneven, the balance of training times of different dependencies is supposed to be guaranteed during training process by the sub-sampling of dependencies. Through several continuous experiments, the following formula of discard probability of high-frequency dependency has been chosen (Mikolov et al. 2013b):

$$P(d_{w_i,w_j}) = 1 - (\sqrt{\gamma} + \gamma + \gamma^2), \tag{15}$$

where $\gamma = \frac{sample}{freq(d_{w_i,w_j})}$, $sample$ is an artificial parameter and $freq(\cdot)$ is the frequency of corresponding dependency. And when $sample = 0$, the sub-sampling is not performed.

The final experiments have proved that the methods outlined above can improve the effectiveness and efficiency significantly[2].

Experiments

In this section, we present several experiments to verify the effectiveness and efficiency of our proposed approach.

Dataset and Training

We trained all embeddings based on partial English Wikipdeia corpus [3], which contains 388,900,648 tokens and 555,434 unique words[4]. To obtain the dependency-based corpus, the initial corpus, containing 5,784 syntactic dependencies and 3.7 million sentences, is parsed by Stanford neural-network dependency parser (Chen and Manning 2014).

[2]Due to the limited space, we only list the most important optimization tricks.

[3]In fact, there are many phrases and abnormal sentences in corpus which cannot be used. So we construct the corpus by limiting the length of sentence (delete the sentences which is too long or too short) and the average length of the sentences after processing is around 15.

[4]The version of download file is wikidata-20161020.

As we found that various dimensions (50, 300, 600, 1000) of word embeddings resulted in similar trends, only experimental results for 300 dimension embeddings will be reported. Meanwhile, we set the dimension of dependency vector $v(d_{w_{i,k-1},w_{i,k}})$ as 50 [5], the initial dependency weight $\varphi_{d_{w_{i,k-1},w_{i,k}}} = 0.9$, and initialize word vector $v(w)$, positive dependency vector $v(d)$ and other model parameters randomly. To remove the fluctuation of experimental results from random initialization, we report the average of several trials for each experiment.

Baseline Methods

We re-implement or use the existing embedding models in the baseline methods as follows:

- GloVe (Jeffrey, Richard, and Christopher 2014): it efficiently leverages global statistical information through factorizing a word-word co-occurrence matrix.
- Original CBOW in Word2vec (Mikolov et al. 2013a): the original CBOW model based on HS.
- Original SG in Word2vec (Mikolov et al. 2013a): the original SG model based on NS.
- Deps (Levy and Goldberg 2014): it improves the original SG based on NS by incorporating first-order dependency into it.
- EXT (Komninos and Manandhar 2016): it introduces second-order dependency into Deps.

Quantitative Results

Now we present the quantitative results of our approach and baselines.

Word Similarity/Relatedness Word similarity/relatedness has been widely used as a task to evaluate the effectiveness of word embeddings. We employ nine datasets, which include SIMLEX-999, TR-3K, WS-353-SIM, VERB-143, MC-30, MTurk-287, RG-65, RW and

[5]Since the dependency vector is used as parameter to assist the prediction model to calculate the probability, we do not introduce it into the evaluation measures of word vectors. Because there are only 5,784 dependencies, too high dimension will only increase the amount of calculation, without obvious effect in the results.

	GloVe	CBOW&HS	SG&NS	Deps	EXT	CBOW$_{Deps}$ &HS (only weight)	SG$_{Deps}$ &NS (only weight)	CBOW$_{Deps}$ &HS	SG$_{Deps}$ &NS
NER	83.64	80.67	82.44	84.07	83.08	81.99	83.89	82.87	**85.14**
LOC	86.43	85.89	87.53	87.68	86.90	86.50	88.75	87.42	**89.63**
MISC	73.77	72.68	75.65	77.52	78.46	74.56	77.53	76.16	**78.79**
ORG	82.36	78.23	79.88	81.87	79.51	80.04	80.85	80.64	**82.53**
PER	88.14	85.86	86.71	89.20	87.42	86.84	88.42	87.27	**89.73**

Table 4: Evaluation results on the test set from the CoNLL-2002 and CoNLL-2003 English data (F1-score).

	GloVe	CBOW &HS	SG&NS	Deps	EXT	CBOW$_{Deps}$ &HS	SG$_{Deps}$ &NS
Prec.	74.3	72.2	74.1	72.4	73.2	73.1	**74.8**
Rec.	67.2	66.5	67.8	65.2	66.2	66.7	**68.1**
F1	70.6	69.2	70.8	68.6	69.5	69.8	**71.3**

Table 5: Evaluation results on the test set from the CoNLL-2012 English data (precision, recall, and F1 of MUC).

YP-130, selected by Manaal and Chris (2014) as the evaluated data. The cosine value of two word vectors is used to measure the degree of similarity/relatedness between them. The Spearman's rank correlation coefficient (Myers and Well. 1995) is used to check the correlation of ranks between human annotation and computed similarities. Our embedding approach manifest its advantages with results of the same method in different context compositions, as shown in Table 3.

Named Entity Recognition (NER) NER is a task to locate and classify words/phrases in sentences into predefined categories such as persons (PER), locations (LOC), organizations (ORG), and miscellaneous (MISC). Meanwhile, NER is also taken as a downstream task to evaluate words embeddings, and we follow the use of word embeddings in NER (Lample et al. 2016). We use CoNLL-2002 and CoNLL-2003 datasets (Sang 2002; Sang and Meulder 2003) that contain independent entity labels for English to train and test our embeddings. As shown in Table 4, the dependency-based word embeddings significantly outperform other baseline methods since the dependency-based context can capture most of stable and valuable syntactic features and partial semantic information.

Neural Co-reference Resolution (NCR) Co-reference Resolution is also an important NLP downstream task. Recently, end-to-end neural network based model has achieved good results through using pre-trained word embeddings based on general methods. For example, Lee et al. (2017) directly considered all spans in a document as potential mentions and learn distributions over possible antecedents for each. Then, we use the English coreference resolution data from the CoNLL-2012 shared task (Pradhan et al. 2012) in Lee et al.'s model to inspect the effect of our embeddings. As shown in Table 5, our methods achieve competitive results compared to other embedding models.

Qualitative Results

Here we present the qualitative results of our approach.

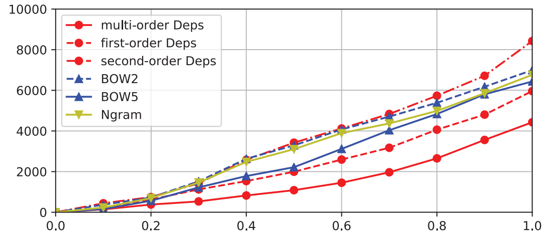


Figure 3: Distribution and quantity of words in composite context based on different methods. The Y-axis denotes the number of unique context words (words have been sorted according their word frequency), the X-axis denotes the ratio between the sum of sorted words' frequency and total word frequency. For example, the line of multi-order dependency in this figure denotes that there are average 4,483 unique words in its corresponding context, and the 2,000 most frequent context words account for about 70% of the total context word frequency.

Context/Compositionality Stability Caused by the various expressions and the complex linguistic phenomena in large-scale corpus, the original composite context used in Word2Vec can be very diverse. To verify this, we randomly choose 5,000 words (with different word frequency) from the vocabulary as examples, and analyze their context compositions acquired from different approaches respectively (such as Word2Vec and etc.). To balance the total frequency of words in each target word's context acquired from different approaches, we standardize the statistic data before analysis. The results are shown in Figure 3. We can see that the multi-order dependency-based method proposed here is able to capture less contextual words than other methods. As shown in Table 6, we select three contrastive words with different word frequencies, and display their top five most frequent words in the context acquired from different methods. We can see that the our extracted words in the context have stronger relatedness with target word.

	N-gram	BOW2	BOW5	1st-Deps	2nd-Deps	mth-Deps
	the	the	the	the	the	21th
	is	was	a	released	released	20th
century	their	studio	was	is	was	dc
	a	debut	released	their	is	bc
	his	an	is	a	recorded	released
	the	the	the	robin	the	robin
	a	robin	a	the	robin	comics
batman	to	is	to	series	a	superman
	was	a	was	superman	is	catwomen
	superman	superman	is	catwoman	series	series
	felicity	felicity	the	felicity	felicity	trees
	are	the	a	trees	was	felicity
huffman	the	was	was	coding	a	coding
	that	are	felicity	born	the	code
	was	coding	is	wilcox	bill	tree

Table 6: The top five frequent words in composite context (based on different strategies) of target words.

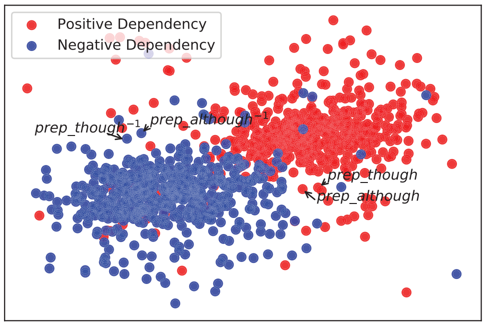


Figure 4: The visualization of dimensionality reduction results of the top 500 highest frequent positive preposition dependencies and their negative dependencies.

Analysis of Dynamic Dependency Parameters

One of the major additional value of this work is that a set of dynamic parameters are used to measure the strength of dependencies between words, which greatly changes the structure of sentence and makes the achievement of the multi-order dependency-based context possible. Hence, words that are really close to target word with respect to syntax and semantics can be selected with only the dependencies among words and target word being concerned. It is easy to find that some dependencies are more important than others through analyzing the experimental results, such as *conj_and*, *conj_or*.

To further analyze the dependencies, we map several trained dependency vectors to low dimensional space by utilizing the most commonly used dimensionality reduction method Principal Component Analysis (PCA) (Jolliffe 1986). Since the number of principal components is limited to two, the dependency vector compressed by PCA algorithm only saves less than 60% of the variance, which means that a small amount of information is lost. However, we can still observe some special characteristics from Figure 4. Due to we set the initial negative dependency vector to 180 degrees form the random initial positive dependency, the trained vectors still roughly retains this relationship.

In addition, as shown in Figure 4, the analogy between words also exists between dependencies, especially between corresponding positive and negative dependencies. Moreover, the cosine similarity between dependencies can also reflect the correct semantic and syntactic relationships between them to a certain extent (for example, the cosine similarity between $prep\_though$ and $prep\_although$ is about 0.81).

Conclusion

This paper introduces an approach which can incorporate the multi-order dependency-based context into original word embeddings with adaptive dependency weights. The representations of dependencies are also integrated into the objective functions to assist the prediction during the training process. Meanwhile, the objective functions, input layer and stochastic gradient based update functions of Word2Vec are modified to adapt to the multi-order dependency-based context. Experiments have shown that the proposed methods not only achieve significant improvement in both direct word similarity/relatedness tasks and indirect downstream NLP tasks, but also solve the problems of original word embeddings (i.e., composite context diversity and relationship confusion), especially in large-scale corpus. Nevertheless, whether our trained dependency parameters can be used to improve the effect of syntax parse or not has not been tested yet and it will be explored in future. Apart from this, the explicit semantic meaning of different dimensions of the dependencies embeddings is still not clear. In the future, it would be interesting to develop a model to explain or visualize the meanings of the vector elements.

Acknowledgement

Jianxin Li is the corresponding author. This work is supported by China 973 Fundamental R&D Program (No.2014CB340300), NSFC program (No.61472022, 61421003, 61772151), SKLSDE-2016ZX-11 and partly by the Beijing Advanced Innovation Center for Big Data and Brain Computing. Yangqiu Song was supported by China 973 Fundamental R&D Program (No. 2014CB340304) and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. 26206717). We also thank the anonymous reviewers for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Baroni, M.; Dinu, G.; and Kruszewski, G. 2014. Don't count, predict! a systematic comparison of context-counting vs context-predicting semantic vectors. In *ACL*, 238–247.

Bengio, Y.; Courville, A. C.; and Vincent, P. 2013. Representation learning: A review and new perspectives. *IEEE Trans. PAMI* 35(8):1798–1828.

Chen, D., and Manning, C. D. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, 740–750.

Chen, K.; Zhao, T.; Yang, M.; and Liu, L. 2017. Translation prediction with source dependency-based context representation. In *AAAI*, 3166–3172.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Eriguchi, A.; Hashimoto, K.; and Tsuruoka, Y. 2016. Tree-to-sequence attentional neural machine translation. In *ACL*, 823–833.

Hegde, C.; Indyk, P.; and Schmidt, L. 2015. A nearly-linear time framework for graph-structured sparsity. In *Proceedings of ICML*, 928–937.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, 873–882.

Jeffrey, P.; Richard, S.; and Christopher, M. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–43.

Jie, Z.; Muis, A. O.; and Lu, W. 2017. Efficient dependency-guided named entity recognition. In *Proceedings of AAAI*, 3457–3465.

Jolliffe, I. T. 1986. Principal component analysis and factor analysis. In *Principal component analysis*. 115–128.

Komninos, A., and Manandhar, S. 2016. Dependency based embeddings for sentence classification tasks. In *NAACL*, 1490–1500.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. In *Proceedings of HLT-NAACL*, 260–270.

Lee, K.; He, L.; Lewis, M.; and Zettlemoyer, L. 2017. End-to-end neural coreference resolution. In *Proceedings of EMNLP*, 188–197.

Levy, O., and Goldberg, Y. 2014. Dependency-based word embeddings. In *ACL*, 302–308.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. In *Proceedings of TACL*, 211–225.

Manaal, F., and Chris, D. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of ACL: System Demonstrations*.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *ICLR*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

Myers, J. L., and Well., A. D. 1995. *Research Design & Statistical Analysis*. Routledge.

Pradhan, S.; Moschitti, A.; Xue, N.; Uryupina, O.; and Zhang, Y. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Proceedings of the Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, EMNLP-CoNLL 2012*, 1–40.

Qiu, L.; Cao, Y.; Nie, Z.; and Rui, Y. 2014. Learning word representation considering proximity and ambiguity. In *ACL*, 1572–1578.

Qiu, L.; Zhang, Y.; and Lu, Y. 2015. Syntactic dependencies and distributed word representations for chinese analogy detection and mining. In *EMNLP*, 2441–2450.

Roy, S., and Roth, D. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of AAAI*, 3082–3088.

Sang, E. F. T. K., and Meulder, F. D. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning*, 142–147.

Sang, E. F. T. K. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning*, 155–158.

Turian, J. P.; Ratinov, L.-A.; and Bengio, Y. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*, 384–394.

Xiang, Y.; Zhou, X.; Chen, Q.; Zheng, Z.; Tang, B.; Wang, X.; and Qin, Y. 2016. Incorporating label dependency for answer quality tagging in community question answering via cnn-lstm-crf. In *Proceedings of COLING*, 1231–1241.

Yin, Y.; Wei, F.; Dong, L.; Xu, K.; Zhang, M.; and Zhou, M. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of IJCAI*, 2979–2985.