An Adaptive Hierarchical Compositional Model for Phrase Embedding

Bing Li[†], **Xiaochun Yang**[†], **Bin Wang**[†], **Wei Wang**[*§], **Wei Cui**[‡], and **Xianchao Zhang**[+]
[†] School of Computer Science and Engineering, Northeastern University, China
[*] University of New South Wales, Australia
[§] Dongguan University of Technology, China
[‡] College of Electrical Engineering and Automation, Shandong University of Science and Technology
[+] School of Software, Dalian University of Technology, China
[†]libing@stumail.neu.edu.cn, {yangxc, binwang}@mail.neu.edu.cn,
[*]weiw@cse.unsw.edu.au, [‡]cuiwei@sdust.edu.cn, [+]xczhang@dlut.edu.cn

Abstract

Phrase embedding aims at representing phrases in a vector space and it is important for the performance of many NLP tasks. Existing models only regard a phrase as either full-compositional or non-compositional, while ignoring the *hybrid-compositionality* that widely exists, especially in long phrases. This drawback prevents them from having a deeper insight into the semantic structure for long phrases and as a consequence, weakens the accuracy of the embeddings. In this paper, we present a novel method for jointly learning compositionality and phrase embedding by adaptively weighting different compositions using an implicit hierarchical structure. Our model has the ability of adaptively adjusting among different compositions without entailing too much model complexity and time cost. To the best of our knowledge, our work is the first effort that considers hybrid-compositionality in phrase embedding. The experimental evaluation demonstrates that our model outperforms state-of-the-art methods in both similarity tasks and analogy tasks.

1 Introduction

Word and phrase embedding aims at representing words and phrases in a continuous vector space where semantically similar words and phrases are embedded near each other. The embeddings can be used to compute semantic relatedness and feed machine learning systems as the input. It have been proven successful in boosting the generalization performance of many NLP tasks such as machine translation [Mikolov *et al.*, 2013a], syntactic parsing [Socher *et al.*, 2013a], and sentiment analysis [Socher *et al.*, 2013b].

Due to its usefulness, there have been many phrase embedding methods developed in the NLP community, and these methods can be broadly classified into three types: (1) compositional phrase embedding, (2) non-compositional phrase embedding, and (3) distinguish compositional phases from non-compositional phrases. The compositional phrase embedding is based on *Frege's Principle of Compositionality*, stating that the meaning of a complex whole is a function only of the meanings of its syntactic part. For example, the meaning of phrase `North America` is associated with the meanings of both `North` and `America`. Therefore, a phrase embedding is computed by aggregating its individual words' embedding using various composition functions [Mitchell and Lapata, 2010; Socher *et al.*, 2012; Yu and Dredze, 2015]. Such compositional phrase embedding methods are scalable to the number of phrases and could learn fine-grained semantic compositionality. However, they are limited by their inability to represent idiomatic phrases, where the meaning of an idiomatic phrase is *not* associated with its component words, *e.g.*, `new york` has a different meaning with either `new` or `york`.

Another type of embedding methods treats a phrase as a single unit and assign a unique embedding for it [Mikolov *et al.*, 2013b; Zhang *et al.*, 2014]. Such phrase embeddings are commonly referred to as non-compositional embedding. Solely relying on non-compositional embedding will suffer from data-sparsity problem since they do not distinguish non-idiomatic phrases from idiomatic phrases. Thus the accuracy of word embeddings for non-idiomatic phrases or their component words are both decreased. In addition, compare with the compositional phrase embeddings, the non-compositional phrase embeddings require extra vectors for phrases which increase the model complexity significantly.

Recently, new techniques are developed to automatically distinguish compositional phases from non-compositional phrases and learn their embedding individually. For instance, Yazdani *et al.* [2015] used supervised models to detect non-compositional phrase. Hashimoto and Tsuruoka [2016] proposed a model that learn compositionality levels by logistic regression. However, they share a significant drawback that they only stick to either full-compositional embedding or non-compositional embedding, while ignore the fact that phrases could be hybrid-compositional, which means there exist both non-compositional and compositional parts in a phrase. For example, phrase `new york city` should be represented as $new\_york \oplus city$, where the words connected with '_' is a non-compositional unit and '$\oplus$' denotes the composition. Such hybrid-compositionality has not been fully explored by existing methods, which prevents them from having a deeper insight into the semantic structure of long phrases.

In order to address this issue, we propose a novel *hierarchical compositional model* for phrase embedding. The goal of our approach is to provide a joint learning paradigm thereby adaptively weighting different compositions by making the first effort to consider hybrid-compositionality of phrase, while at the same time optimizing its embedding accordingly. In addition to learning phrase embeddings, our approach could also mine valuable phrases and infer their internal structures. In our model, the compositions of a phrase are represented by an implicit hierarchical structure, and phrase embedding are defined as a mixture of non-compositional and the hierarchically compositional embedding. The proportion between the two components are adaptively adjust by composability. In this way, the weight could be decoupled with a specified composition, thus lower the model complexity. In order to avoid unnecessary cost caused by different constituent partition, our model could joint learn model parameters and probabilistically reason about the optimal constituent partition. Our method is purely data-driven, where all the parameters are jointly learned in conjunction with the target task for learning phrase embedding. We evaluated our model on both similarity tasks and analogy tasks, the results demonstrate that our model outperforms state-of-the-art models.

2 Hierarchical Compositional Model

Given a phrase `new york city`, it could have four distinct compositions: new_york_city, $new\_york \oplus city$, $new \oplus york\_city$, and $new \oplus york \oplus city$. We hope to propose a "good" model to get the correct composition of $new\_york \oplus city$ and its resulting embedding in a purely unsupervised manner. To incorporate all the compositions, a straightforward solution is to use weighted sum model and represent phrase embedding as the weighted sum of all its compositions (see Eq. (1)), and learn the right composition by adjusting the weight $g(\cdot)$ of each composition.

$$\boldsymbol{vec}(p) = \sum_{h \in H} g(h) \cdot \boldsymbol{vec}(h), \tag{1}$$

where H is the set of all compositions of a phrase p, and $\boldsymbol{vec}(h)$ is the embedding under composition $h \in H$.

Since there are 2^{n-1} compositions for an n-gram phrase (*i.e.* a phrase with n words), the weighted sum model needs to keep $\mathcal{O}(|Pr| \cdot 2^n)$ free parameters for a corpus with entire phrase set Pr. Such a high model complexity generally results in overfitting on training data.

To tackle this problem, we propose a novel hierarchical compositional (HC) model that can provide lower model complexity to learn better phrase embeddings. In our model, a phrase's embedding $\boldsymbol{vec}(p)$ is defined as a mixture of its non-compositional embedding $\boldsymbol{e}(p)$ and a hierarchically compositional embedding. The latter one is defined on the phrase's optimal constituent partition C_p^* which forms an implicit hierarchical structure, *e.g.*, $new\_york \oplus city$ can be described as a composition of `new york` and `city`, while `new york` is a non-composition of `new` and `york`. The proportion of the two embeddings is adaptively adjust by its composability (the level of compositionality) $\beta(p)$. Formally, phrase embedding $\boldsymbol{vec}(p)$ is computed as follow:

$$\boldsymbol{vec}(p) = (1 - \beta(p)) \cdot \boldsymbol{e}(p) + \beta(p) \cdot \sum_{c \in C_p^*} \boldsymbol{vec}(c), \tag{2}$$

where $\beta(p)$ $(0 \leq \beta(p) \leq 1)$ is a free parameter for a given phrase p to denote its composability, and $\beta(p) = 0$ if p is a word. A larger value of $\beta(p)$ indicates p is of higher composability and *vice versa*. Any constituent $c \in C_p^*$ could be either a single word or a multi-words term (*i.e.* phrase). The constituent partition refers to the non-overlapping sequence partition of a phrase, *e.g.,* $\{$`new york`, `city`$\}$ is a constituent partition of phrase `new york city`, in which `new york` or `city` is a constituent. We will discuss how to infer the optimal constituent partition C_p^* in Section 3.2 and how to learn parameters $\beta(p)$ and $\boldsymbol{e}(p)$ in Section 3.3.

There are at least two reasons why our HC model is superior to the weighted sum model. Firstly, it could be decoupled with a specific composition since the weights of compositions are implicitly represented by aggregating the composability of phrase. Therefore, HC model could provide a high enough representation ability with only $\mathcal{O}(|Pr|)$ parameters, which greatly reduces model complexity. Secondly, the loss can be back-propagated from a phrase to its component words. It could improve both phrase and component word embeddings.

In this paper, we adopt widely used element-wise addition [Tian *et al.*, 2017] as our composition function.

3 Phrase Embedding Based on Hierarchical Compositional Model

Our goal is to learn "good" model parameters ($\beta(p)$ and $\boldsymbol{e}(p)$) with the help of constituent partition to generate better phrase embeddings. In this section, we propose a Hierarchical Compositional based Phrase Embedding (HCPE) approach to joint learning model parameters and optimal constituent partition.

3.1 HCPE Framework

Obviously, constituent partition and phrase embedding could benefit from each other. A "good" constituent partition could construct a correct hierarchical structure to facilitate better phrase embedding. On the other hand, "good" word/subphrase embeddings could also facilitate generating better partition: a better partition is supposed to further improve the prediction of context (or center word) to get better embedding. Therefore, we propose an HCPE approach to joint learning phrase embeddings and constituent partitions.

Let $S = p_1, p_2, ..., p_{|S|}$ be a corpus, in which each p_i could be either a single word or a phrase ($|S|$ denotes the number of p_i in S), V be the set of vocabulary (*i.e.*, distinct words and phrases) in S, and $v_i \in V$ is a distinct word or phrase. P_r be the set of distinct phrases in S, and $pr_i \in Pr$ is a distinct phrase. HCPE aims to maximize the joint probability $P_{\boldsymbol{\theta}}(S, \mathbb{C})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \ \max_{\mathbb{C}} P_{\boldsymbol{\theta}}(S, \mathbb{C}), \tag{3}$$

where $\mathbb{C} = \{C_{pr_1}, C_{pr_2}, ..., C_{pr_{|Pr|}}\}$ is a set of latent variables in which each C_{pr_i} denotes a latent variable to denote the constituent partition of phrase pr_i, and $\boldsymbol{\theta} = \{\boldsymbol{e}(v_1), \ldots, \boldsymbol{e}(v_{|V|}), \beta(pr_1), \ldots, \beta(pr_{|Pr|})\}$ is the model parameter.

In this paper, we adopt commonly used Skip-Gram [Mikolov *et al.*, 2013b] as our embedding architecture. Based on Skip-Gram [1], we have:

$$\begin{aligned} P_{\boldsymbol{\theta}}(S, \mathbb{C}) &= P_{\boldsymbol{\theta}}(S|\mathbb{C}) \cdot P(\mathbb{C}) \\ &= \prod_{i=1}^{|S|} \prod_{u \in Cont(p_i)} P_{\boldsymbol{\theta}}(u|p_i, C_{p_i}) \cdot P(C_{p_i}), \end{aligned} \tag{4}$$

where $Cont(p_i)$ is the set of context words/phrases of p_i, $C_{p_i} \in \{C_{p_i}^1, C_{p_i}^2, ..., C_{p_i}^{2^{|p_i|-1}}\}$, in which $C_{p_i}^j$ denotes a specific constituent partition of p_i, *e.g.*, {`new`, `york city`}, {`new york`, `city`}, or {`new`, `york`, `city`} for phrase `new york city`. Note that, if p_i is a single word, we have $P(C_{p_i}) = 1$ and $P(u|p_i, C_{p_i}) = P(u|p_i)$.

With the form of Eqs. (3) and (4), maximizing the joint probability in Eq. (3) is actually a Hard-Expectation-Maximization (Hard-EM) problem [MacKay, 2003], which has been proved to get converged and achieve a local optimum by a coordinate ascent strategy [MacKay, 2003]. To infer latent variables $\mathbb{C}$ and learn model parameters $\boldsymbol{\theta}$, the following two steps are repeatedly performed:

- E-step (Expectation step): inferring the optimal constituent partition $C_{p_i}^*$ of a phrase p_i with a fixed $\boldsymbol{\theta}$.

$$C_{p_i}^* = \arg\max_{C_{p_i}} P_{\boldsymbol{\theta}}(u|p_i, C_{p_i}) \cdot P(C_{p_i}); \tag{5}$$

- M-step (Maximization step): learning model parameters using $C_{p_i}^*$ inferred by E-step.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(u|p_i, C_{p_i}^*). \tag{6}$$

3.2 E-step: Inferring Optimal Constituent Partition

Firstly, we propose a generative model to compute the probability of generating a constituent partition (*i.e.*, $P(C_{p_i})$). Secondly, considering that the enumerating-based method is intractable to to infer optimal constituent partition, we propose a dynamic programming method to make the computation feasible.

Probability of Constituent Partition

Given an n-gram phrase $p = w_1 \ldots w_n$, let $C_p = \{c_1, \ldots, c_m\}$ be any constituent partition of p, which can be separated by a set of split boundaries $B = \{b_1, \ldots, b_m, b_{m+1}\}$, where $b_1 = 1$ and $b_{m+1} = n + 1$.

$$P(C_p) = \prod_{c_i \in C_p} P(\rho, c_i, b_{i+1}|b_i) \tag{7}$$

where $P(\rho, c_i, b_{i+1}|b_i)$ is the conditional probability of observing c_i is the i-th constituent, and ρ is a phrase indicator.

Eq. (7) is derived from the following generative model: Firstly, we generate split boundary b_{i+1} with a probability of $P(L)$ assuming the order-1 Markov property, where L is a random variable and $P(L)$ is the length prior over the number of words in c_i, $P(L)$ can be computed by:

$$P(L) \propto \alpha^{1-L}, \tag{8}$$

where $\alpha \leq 1$ is a length penalty factor. A smaller α rewards a longer constituent length which would make the hierarchical structure deeper and thus provide a more plentiful compositional diversity.

Secondly, we generate c_i according to the probability

$$P(c_i|L = l(c_i)) = \frac{f(c_i)}{\sum_{\forall l(c_j)=l(c_i)} f(c_j)}, \tag{9}$$

where $f(c_i)$ denotes the frequency of c_i in the corpus. This probability is based on the frequent term have high probability to be a whole unit.

Thirdly, we generate the phrase indicator ρ of c_i. If c_i is a phrase in the vocabulary V (including all words and phrases) , it should be a strong evidence that the whole c_i is a unit. Therefore, we exclude the word sequences which is not in V by:

$$P(\rho|c_i) = \begin{cases} 1, & if\ c_i \in V \\ 0, & otherwise. \end{cases} \tag{10}$$

Based on the above, $P(\rho, c_i, b_{i+1}|b_i)$ in Eq. (7) can be probabilistically factorized as:

$$\begin{aligned} P(\rho, c_i, b_{i+1}|b_i) &= P(b_{i+1}|b_i) \cdot P(c_i|b_{i+1}, b_i) \cdot P(\rho|c_i) \\ &= P(L) \cdot P(c_i|L{=}l(c_i)) \cdot P(\rho|c_i). \end{aligned} \tag{11}$$

Efficiently Inferring Optimal Constituent Partition

Taken *softmax* as the activation function, the joint probability can be represented as:

$$\begin{aligned} &P(u|p_j, C_{p_j}) \cdot P(C_{p_j}) = \frac{\mathrm{e}^{\boldsymbol{e}(u)^\top \boldsymbol{vec}(p_j)}}{\sum_{v \in V} \mathrm{e}^{\boldsymbol{e}(v)^\top \boldsymbol{vec}(p_j)}} \cdot P(\rho, c_i, b_{i+1}|b_i) \\ &= \frac{\mathrm{e}^{(1-\beta(p_j))\boldsymbol{e}(u)^\top \boldsymbol{e}(p_j)}}{\sum_{v \in V} \mathrm{e}^{\boldsymbol{e}(v)^\top \boldsymbol{vec}(p_j)}} \prod_{c_i \in C_{p_j}} \mathrm{e}^{\beta(p_j) \cdot \boldsymbol{e}(u)^\top \boldsymbol{vec}(c_i)} P(\rho, c_i, b_{i+1}|b_i) \end{aligned} \tag{12}$$

Finally, based on Eq. (12), E-step is to solve the following optimization problem:

$$C_{p_j}^* = \arg\max_{C_{p_j}} \prod_{c_i \in C_{p_j}} \mathrm{e}^{\beta(p) \cdot \boldsymbol{e}(u)^\top \boldsymbol{vec}(c_i)} P(\rho, c_i, b_{i+1}|b_i). \tag{13}$$

Intuitively, a straight-forward method is to enumerate every possible constituent partition to find the best solution, which is intractable ($\mathcal{O}(2^n)$ complexity) in practice. Therefore, the

[1] Other architectures like CBOW [Mikolov *et al.*, 2013b], GloVe [Pennington *et al.*, 2014], *etc* can be readily applied.

challenge here is how to avoid enumerating constituent partitions to make computation efficient.

We propose a dynamic programming-based strategy to find the optimal constituent partition to maximize Eq. (13) while keep the computation tractable. It is based on the observation that if $C^*_{[1,i]}$ is the optimal constituent partition of phrase $w_1 \dots w_{i-1}$, then there must have a constituent $c_{[k,i]} \in C^*_{[1,i]}$ ($k < i$), makes $C_{[1,i]} - \{c_{[k,i]}\}$ be the optimal constituent partition of phrase $w_1 \dots w_{k-1}$. Therefore, we set up a vector $\boldsymbol{M} \in [0,1]^{n+1}$, in which each component M_i ($i \in \{1, \dots, n+1\}$) stores the optimal probability of phrase $w_1 \dots w_{i-1}$. Initially, we have $M_1 = 1$. The recursion function is given as follows:

$$M_i = \max_{k\in[1,i-1]} \; M_k \cdot \mathrm{e}^{\beta(p_j)\cdot \boldsymbol{e}(u)^\top \boldsymbol{vec}(c_i)} \cdot P(\rho, c_i, i|k). \qquad (14)$$

Based on Eq. (14), we choose the appropriate k to make M_i maximal. The value stored in M_{n+1} equals the maximal value of Eq. (13). Then the optimal constituent set can be easily fetched by back-tracking the vector from M_{n+1}. Since we need to fill an $(n+1)$-length vector, and the cost of computing each cell is $\mathcal{O}(n)$, the total time complexity is $\mathcal{O}(n^2)$.

3.3 M-step: Learning Model Parameters

Now we show how to learn model parameters $\beta(p)$ and $\boldsymbol{e}(p)$ based on the optimal constituent partition. We adopt stochastic gradient descent (SGD) to optimize the objective function using random initialization. Considering that the original *softmax* function is impractical because its huge compuational cost [Mikolov *et al.*, 2013b], in practice, some approximated objective fucntion are used to speed up the processing. We use J to denote a given approximated objective function, *e.g.,* Hierarchical Softmax, Negative Sampling [Mikolov *et al.*, 2013b], *etc*. Let $\boldsymbol{\delta}^{(l)}_p \in \mathbb{R}^{d\times 1}$ (d is the dimension of embedded vectors) be the partial derivative of J with respect to the variable $\boldsymbol{vec}(p)$ on l-th layer of the hierarchical structure. Notice that, if $l = 1$, we have $\boldsymbol{\delta}^{(1)}_p = \frac{\partial J}{\partial \boldsymbol{vec}(p)}$. The partial derivative of J w.r.t. any lower-layer constituent c_i can be computed via its direct upper-layer phrase p's partial derivative according to the following equation:

$$\boldsymbol{\delta}^{(l+1)}_c = \beta(p) \cdot \boldsymbol{\delta}^{(l)}_p. \qquad (15)$$

In this way, the error term can be back-propagated via the hierarchical structure.

At each layer, the gradient of non-compositional embedding $\boldsymbol{e}(p)$ of phrase p can be computed by:

$$\frac{\partial J}{\partial \boldsymbol{e}(p)} = (1 - \beta(p)) \cdot \boldsymbol{\delta}_p. \qquad (16)$$

From Eqs. (15) and (16), we can see that β is used for adjusting the error term's proportion between the non-compositional embedding and compositional embeddings. If β is close to 0, the non-compositional embedding are mainly updated, and *vice versa*.

At each layer, the partial derivative of $\beta(p)$ can be computed by the follow equation:

$$\frac{\partial J}{\partial \beta(p)} = \boldsymbol{\delta}(p)\cdot \frac{\partial \boldsymbol{vec}(p)}{\partial \beta(p)} = \boldsymbol{\delta}(p)\cdot(-\boldsymbol{e}(p)+ \sum_{c_i\in C_p} \boldsymbol{vec}(c_i)). \qquad (17)$$

Eq. (17) shows that, $\beta(p)$ is used for adjusting between non-compositional embedding $\boldsymbol{e}(p)$ and hierarchical compositional embedding $\boldsymbol{vec}(p)$. To be specific, if the error term $\boldsymbol{\delta}(p)$ is more closer to compositional embedding than the non-compositional embedding, *i.e.*, $\boldsymbol{\delta}(p) \cdot \sum_{c_i\in C_p} \boldsymbol{vec}(c_i) > \boldsymbol{\delta}(p) \cdot \boldsymbol{e}(p)$, $\beta(p)$ needs to be bigger accordingly, and *vice versa*. The intuition behind this is that: (i) if a phrase is non-compositional, its contexts is often different from the contexts of its constituents; and (ii) if a phrase is compositional, its contexts is similar with the contexts of its constituents, *i.e.*, its constituents and the contexts co-occurs frequently.

From both human judgments and empirical results, β follows a U-sharp distribution indicating the composability of most of phrases are very close to either 0 or 1. Moreover, it should be cut-off when β reaches either 0 or 1 to avoid overflowing. Therefore, we need a large learning rate when β is near 0.5 , and gradually decrease the learning rate towards 0 or 1. For $\beta(p)$, we use the following learning ratio η_p:

$$\eta_p \propto \beta(p) \cdot (1 - \beta(p)). \qquad (18)$$

In our HC model, the error term are back-propagated recursively from the root phrase to each non-composable unit until every composability ratio β and non-compositional embedding $\boldsymbol{e}$ are updated, which helps to improve both phrase embeddings and constituents' embeddings.

3.4 Complexity Analysis

Model complexity: Besides the conventional embedding model (*e.g.,* Skip-Gram, CBOW), our model requires $\Theta(|Pr|)$ parameters for a set of phrases Pr. Taken Skip-Gram as our embedding architecture, then the overall model complexity would be $\Theta(|V| \cdot T + |Pr|)$, where $|V|$ is the vocabulary size and T is the dimension of vector. Generally, $|V| \cdot T \gg |Pr|$, thus our model has almost the same model complexity with conventional embedding models.
Time complexity: For an n-gram phrase p, the cost of E-step is $\mathcal{O}(n^2)$, and the cost of M-step is $\mathcal{O}(n)$. Since E-step and M-step are alternately performed to scan the whole corpus S, the total time complexity is $\mathcal{O}(n^2|S|)$. Considering that a phrase commonly keeps a relative short and fixed length (among 2-6), the n^2 term can be regarded as a constant value ω, and thus the complexity is $\mathcal{O}(\omega|S|)$.

4 Experimental Evaluation

In this section, we evaluated our HCPE model on similarity tasks and analogical reasoning tasks, and demonstrate its superiority over state-of-the-art models. Further, a qualitative analysis along with a case study show the capability of HCPE to identify the composability of phrase embeddings.

4.1 Training Corpora

We used the following three training corpora:

- *Text8*[2] contains the first billion characters extracted from English Wikipedia.
- *GoogleNews*[3] contains all Google news articles in 2012.
- *Wiki*[4] contains all English Wikipedia articles in 2015.

The detailed statistics are summarized in Table 1.

Data sets	*Text8*	*GoogleNews*	*Wiki*
Data Size	100M	1.8G	8.9G
# of Words	17 million	0.28 billion	2 billion
# of Vocabularies	71K	681K	1233K

Table 1: Statistics on the three datasets.

4.2 Compared Models

To demonstrate the effectiveness of our model, we compared our model with the following state-of-the-art models:

- Word2vec [Mikolov *et al.*, 2013b] is a two-layer neural network model to produce words or phrases embeddings. It take all phrases as non-compositional. Here we use its Skip-Gram with Negative Sampling model.
- SVO [Hashimoto and Tsuruoka, 2016] learns a scoring function to detect compositional and non-compositional in phrase embedding. We adopt this method as it represent the state-of-the-art method for jointly learning compositional and non-compositional phrase embedding.

Kartsaklis *et al.* [2014] reported that non-compositional embeddings are better than full-compositional embeddings in phrase similarity task. Therefore, we do not compare with a full-compositional method.

4.3 General Experimental Setting

Our HCPE approach adopts Skip-Gram architecture, as it represents the state-of-the-art embedding model.

For optimization, both state-of-the-art methods and our method used Negative Sampling (NEG) technique [Mikolov *et al.*, 2013b] to speed up training, and all methods use the same vector dimension and default setting. To be specific, we set the number of negative examples to be 25, and iterations (number of epochs) to be 5. The initial learning rate of Skip-Gram model were set to 0.05. We set the dimension of vector $d = 200$, unless noted otherwise. We set context window length to be 10 and sub-sampling rate $1\mathrm{e}-5$.

In order to properly initialize parameter β, we had tried several different strategies, including: (1) Randomly initializing β by a Uniform distribution $U[0,1]$; (2) Randomly initializing β by a Gaussian distribution $N(0.5, 1)$; (3) Initializing β by a fixed value, *e.g.*, let $\beta = 0.5$ for all phrases; (4) Initializing β by their phrase's normalized PMI (NPMI), *i.e.*, $\beta = 1 - npmi$. According to experimental tests, random Gaussian was the best initialization strategy (was 3.78% better than random Uniform, 5.75% better than the best fixed β (0.5), and was 7.85% better than NPMI, on phrase analogical reasoning tasks). Thus, we randomly initialized β by a Gaussian distribution $N(0.5, 1)$.

To validate the effectiveness of our learning rate of $\beta(p)$, we compared with two conventional approaches: (1) fixed learning rate, *i.e.*, $\eta_p \in \{0.025, 0.05, 0.1\}$; and (2) add a regularization term $||\beta(p)^2 + (1-\beta(p))^2||$on $\beta(p)$. We tested them on three corpora, on average, our approach was 1.20% better than the best fixed learning rate (0.05), and was 8.48% better than regularization approach on phrase analogy tasks.

4.4 Similarity Tasks

In the similarity tasks, each model needs to compute semantic similarity (*Cosine* similarity) for any given word/phrase pair, and the correlation between the result and gold standard are reported as the model performance. We adopt two correlation measurements: Spearman correlation ρ and Pearson correlation γ. We tested our model on both phrase and word similarity tasks with the same trained vectors to demonstrate our model could effectively reduce the sparsity problem and could help not only phrase similarity but also word similarity.

Phrase similarity

For phrase similarity tasks, we adopt three test datasets: (1) **TR9856** [Levy *et al.*, 2015] contains human-labeled phrase-relatedness values for 9856 phrase pairs. (2) **PhraseSim** [Mitchell and Lapata, 2010] contains human-rated similarity at the scale of 0 to 9 for 324 phrases pairs. (3) **WP-300** [Li *et al.*, 2013] contains 200 phrase pairs and 100 word pairs with human-assigned similarity judgment.

Corpus	Model	**TR9856**		**PhraseSim**		**WP-300**	
		ρ	γ	ρ	γ	ρ	γ
Text8	HCPE	**0.598**	**0.600**	0.450	0.509	**0.740**	**0.747**
	Word2vec	0.579	0.576	0.414	0.311	0.653	0.676
	SVO	0.556	0.563	**0.487**	**0.525**	0.733	**0.747**
GoogleNews	HCPE	**0.623**	**0.620**	**0.623**	**0.647**	0.533	**0.567**
	Word2vec	0.578	0.571	0.352	0.315	0.446	0.528
	SVO	0.617	0.618	0.608	0.613	**0.561**	0.565
Wiki	HCPE	0.626	**0.653**	**0.802**	**0.933**	**0.741**	**0.750**
	Word2vec	**0.644**	0.635	0.651	0.625	0.546	0.619
	SVO	0.632	0.649	0.800	0.915	0.646	0.606

Table 2: Results on the phrase similarity tasks, the best results are marked in bold font.

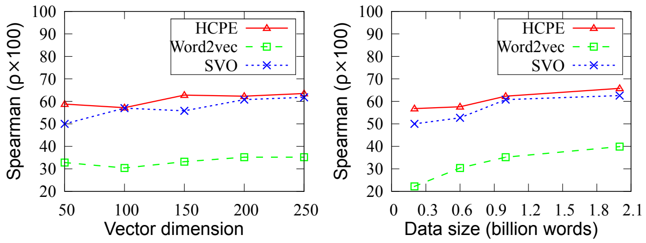
Table 2 shows results of HCPE, Word2vec, and SVO trained on three corpora and tested using the above three test datasets. HCPE achieved a better performance in most cases. An interesting observation is that, on *Text8* corpus, HCPE performed sightly better than others on **TR9856** while little worse than SVO on **PhraseSim** dataset. The reason is that, *Text8* is a very small corpus (only 17 million words), it is hard for parameter β in HCPE to get converged in such a small corpus, thus it inevitably involves error at the unconverged period. But once it got a large enough corpus (*e.g.*, *GoogleNews* or *Wiki*), HCPE performed predominately better than Word2vec and SVO.

We also investigated the influence of data size and vector dimension on phrase embeddings, as shown in Figure 1. For testing the influence of data size, we varied data size from 0.2 billlion to 2 billion by increasing the iteration on *GoogleNews* dataset, and reported the Spearman correlation on **Phrasesim** dataset. Figure 1(a) shows that, all models could gain better results with a larger data size. Compared with Word2vec,

[2] https://cs.fit.edu/ mmahoney/compression/textdata.html

[3] http://www.statmt.org/wmt14/training-monolingual-news-crawl/news.2012.en.shuffled.gz

[4] https://dumps.wikimedia.org/



(a) Varying data sizes.

(b) Varying dimensions.

Figure 1: The results of phrase similarity task.

HCPE can quickly achieve a better result with a small amount of data size (less than 0.6 billion words), while Word2vec could achieve only 0.30 correlation. Compared with SVO, our method could gain a better result in any data size. Figure 1(b) shows the result when increasing the vector dimension. We can see that different dimension settings (from 50 to 250) do not cause significant changes for the three models, and our model is still the best among the three.

Word similarity

We adopt three test datasets: (1) **SimLex-999**[5] is a gold standard resource for the evaluation of models that learn the meaning of words. (2) **Sim-353**[6] contains 353 word pairs along with human-assigned similarity judgments. (3) **WP-130** [Li *et al.*, 2013] contains 130 word/phrases pairs along with human-assigned similarity judgments.

Corpus	Model	**SimLex-999**		**Sim-353**		**WP-300**	
		ρ	γ	ρ	γ	ρ	γ
Text8	HCPE	**0.323**	0.336	**0.647**	**0.641**	**0.608**	0.590
	Word2vec	0.321	**0.338**	0.636	0.618	0.585	**0.595**
	SVO	0.300	0.321	0.597	0.581	0.598	0.593
GoogleNews	HCPE	**0.374**	**0.366**	**0.673**	**0.650**	**0.561**	**0.529**
	Word2vec	0.285	0.304	0.636	0.622	0.496	0.497
	SVO	0.356	0.366	0.633	0.622	0.522	0.514

Table 3: Results on the word similarity task.

Table 3 shows the results of word similarity tasks. As expected, our HCPE model achieved a better performance, since it could adaptively adjust between different compositions, and update constituent words' embeddings by backpropagating error from its phrase embedding. Thus, compared with non-compositional model Word2vec, and binary compositional model SVO, our model could greatly reduce the sparsity caused by phrases so that it could improve both phrase embeddings and word embeddings.

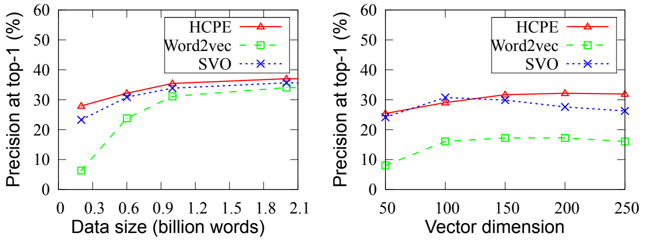
4.5 Analogical Reasoning Task

We adopt **Google semantic analogy** dataset introduced by Mikolov *et al.* [2013b] which contains 3218 phrase analogy questions among five relations. Here we use the precision@top-1 as measurement, which means for each question the model is only allowed to return its top-1 similar phrase as answer. Table 4 reports the results of the three phrase embedding models.

Corpus	Dimension	HCPE	Word2vec	SVO
Text8	50	24.95	8.05	24.16
	100	29.23	16.09	30.76
	200	**32.18**	17.24	27.59
GoogleNews	50	33.81	16.45	31.09
	100	**37.34**	16.09	35.36
	200	35.37	31.14	33.87

Table 4: Precision@top-1 on analogical reasoning task.

Table 4 shows the performance of Word2vec is the worst among the three, especially on small corpus, indicating that non-compositional phrase embedding models are easier to suffer from sparsity problem. HCPE could achieve a better performance than SVO nearly in all settings. This demonstrates comprehensively incorporating hybrid-composition, full-composition and non-composition using our hierarchical compositional structure is superior to the binary compositional model and account for its better performance.



(a) Varying data sizes.

(b) Varying dimensions.

Figure 2: The results of analogical reasoning task.

We also compared the performance of HCPE model with Word2vec and SVO in analogy tasks with various corpus sizes and vector dimensions. Figure 2(a) shows the precision when increasing the corpus size from 0.2 billion to 2 billion. We can see the that, for all the three models, the larger corpus is used, the better the performance will be. Compared with Word2vec and SVO, our model could achieve the best precision in all corpus size settings. To be specific, our model could quickly achieve much higher precision than Word2vec and SVO on both small and large corpora. Figure 2(b) shows the result on various vector dimensions on *Text8* corpus. Intuitively, dimensionality directly relates to the number of parameters to learn, and that's why we demonstrate its impact of these algorithms for varying dimensionality. Our model performs significantly better than Word2vec in all dimensions but performs slightly worse than SVO on 100 vector dimension, but on the whole, our model performs better on other settings and achieves best precision (32.18%).

4.6 Phrase Composability Analysis

To investigate the quality of composability β, we randomly sampled 104 phrases and asked three knowledgeable people to rate on a scale of 0 – 10 to judge the degree of a phrase's composability, based on criteria that whether the phrase has a figurative meaning that is different from its literal meaning, or the phrase has a meaning that is not a simple composition of the meanings of its constituent words [Mikolov *et al.*, 2013b]. Table 6 shows the proportions of different compositionalities.

[5] http://www.cl.cam.ac.uk/ fh295/simlex.html

[6] http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/

Phrase	Compositionality	Highest-weighted composition	Corresponding weight	Human-rated weight
`salt lake city`	Non-compositional	$salt\_lake\_city$	1.0	1.0
`new york times`		new_york_times	1.0	1.0
`human rights watch`		$human\_rights\_watch$	1.0	0.93
`new york stock exchange`	Hybrid-compositional	$new\_york \oplus stock \oplus exchange$	0.99	0.83
`united states air force`		$united\_states \oplus air\_force$	0.99	1.0
`vice president joe biden`		$vice\_president \oplus joe\_biden$	0.99	1.0
`european financial stability`	Full-compositional	$european \oplus financial \oplus stability$	0.98	0.93
`international space station`		$international \oplus space \oplus station$	0.98	0.73
`high blood pressure`		$high \oplus blood \oplus pressure$	0.99	0.76

Table 5: Case study using our HCPE.

Corpus	Non-comp.	Hybrid-comp.	Full-comp.
Text8	8.6%	11.8%	79.5%
GoogleNews	3.3%	8.5%	88.1%
Wiki	7.9%	12.5%	79.6%

Table 6: The proportions of phrase composabilities.

We averaged the three human ratings as gold standard and computed the Spearman ρ and Pearson γ correlation between the gold standard and the result of our model. For comparison, we also lists the correlation of three fixed β settings $\beta \in \{0, 0.5, 1\}$. Table 7 shows the result of phrase composability analysis.

	$\beta = 0$	$\beta = 0.5$	$\beta = 1$	HCPE
ρ	-0.14899	0.18217	0.16295	**0.48324**
γ	-0.14582	0.14764	0.15625	**0.60912**

Table 7: Correlation of phrase composability.

HCPE performs significantly better than the three fixed β settings, which demonstrate the effectiveness of our model.

4.7 Case Study

We also provide a case study based on the result of HCPE model on *GoogleNews* dataset. Table 5 lists 9 sampled phrases and classifies them into three groups (non-compositional, hybrid-compositional, and full-compositional) according to their compositionality. The weight of each composition is computed by multiplicating composability of generating this composition. For each phrase, we show its highest-weighted composition inferred by HCPE along with a human rated weight about it.

From the table we can see that, the highest-weighted composition of each phrase is in accordance with human intuition and the weight is very close to human judgment. The non-compositional phrase (*e.g.*, `salt lake city`) or non-compositional constituent (*e.g.*, `new york`) are correctly picked up by HCPE model. Furthermore, our model could flexibly adjust between different compositionality and adaptively learn the right composition. This demonstrate HCPE model is of high effectiveness and works well in practice.

5 Related Work

The importance of phrase embedding has led to many researches over the past few years. Existing works can be broadly classified into three categories according to their treatment of compositionality.

The first category represents a phrase's embedding as the aggregation of its individual word's embedding on top of various composition functions, such as vector addition/averaging [Tian *et al.*, 2017], component-wise multiplication, and tensor products [Mitchell and Lapata, 2010]. There are complicated functions such as weighted linear compositional model [Yu and Dredze, 2015], ordered tensor products [Clark and Pulman, 2007; Coecke *et al.*, 2010; Socher *et al.*, 2013a], and recursive matrix-vector neural network [Baroni and Zamparelli, 2010; Socher *et al.*, 2012].

The second category treats phrases as single units, and assigns a unique embedding for each phrase. Mikolov *et al.* [2013b] models phrase as a single unit. Levy and Goldberg [2014] and Yin *et al.* [2016] train non-compositional phrase embedding using dependency parsing. Yazdani *et al.* [2015] adopt a supervised model to detect non-compositional phrases. Katz and Giesbrecht [2006] adopt LSA to automatically identify non-compositional phrases. Albeit non-compositional embeddings are preferable when dealing with idiomatic phrases, it may suffer from data-sparsity problem.

The third category tries to distinguish compositional and non-compositional phrase. Farahmand *et al.* [2015] provide compositionality dataset for compound nouns. McCarthy *et al.* [2007] and Venkatapathy and Joshi [2005] use distributional features of corpus to detect compositionality. Kiela and Clark [2013] detect the compositionality using word embeddings. Hashimoto and Tsuruoka [2016] learns compositionality by a scoring function. Wang *et al.* [2017] propose a supervised model to learn sentence-level embedding. In their model, a sentence is always compositional, while a constituent's embedding could be the max-pooling of its compositional and non-compositional embeddings.

Existing models only stick to either full-composition or non-composition, while ignore the hybrid-composition of phrases. By contrast, our model could adaptively learn phrase compositionality and updating its embedding accordingly.

6 Conclusion

In this paper, we proposed a HCPE method to learn phrase embedding. Compared with the best state-of-the-art method SVO, on average, our method was 5.5% better in analogical reasoning tasks and was 1.8% better in phrase similarity task.

Acknowledgments

The work was partially supported by the National Natural Science Foundation of China (Nos. U1736104, 61572122, 61532021), CSC scholarship No. 201606080078. Wei Wang was partially supported by D2DCRC DC25002 and DC25003, and ARC DP 170103710 and 180103411. Wei Cui was supported by Natural Science Foundation of Shandong Province No. ZR201709250069. Xiaochun Yang is the corresponding author of this work.

References

[Baroni and Zamparelli, 2010] Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *EMNLP*, pages 1183–1193. ACL, 2010.

[Clark and Pulman, 2007] Stephen Clark and Stephen Pulman. Combining symbolic and distributional models of meaning. In *AAAI Spring Symposium*, 2007.

[Coecke *et al.*, 2010] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

[Farahmand *et al.*, 2015] Meghdad Farahmand, Aaron Smith, and Joakim Nivre. A multiword expression data set: Annotating non-compositionality and conventionalization for english noun compounds. In *MWE@ NAACL-HLT*, pages 29–33, 2015.

[Hashimoto and Tsuruoka, 2016] Kazuma Hashimoto and Yoshimasa Tsuruoka. Adaptive joint learning of compositional and non-compositional phrase embeddings. In *ACL*, pages 205–215, 2016.

[Kartsaklis *et al.*, 2014] Dimitri Kartsaklis, Nal Kalchbrenner, and Mehrnoosh Sadrzadeh. Resolving lexical ambiguity in tensor regression models of meaning. In *ACL*, pages 212–217, 2014.

[Katz and Giesbrecht, 2006] Graham Katz and Eugenie Giesbrecht. Automatic identification of non-compositional multi-word expressions using latent semantic analysis. In *COLING*, pages 12–19, 2006.

[Kiela and Clark, 2013] Douwe Kiela and Stephen Clark. Detecting compositionality of multi-word expressions using nearest neighbours in vector space models. In *EMNLP*, pages 1427–1432, 2013.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, pages 302–308, 2014.

[Levy *et al.*, 2015] Ran Levy, Liat Ein-Dor, Shay Hummel, Ruty Rinott, and Noam Slonim. Tr9856: A multi-word term relatedness benchmark. In *ACL*, pages 419–424, 2015.

[Li *et al.*, 2013] Peipei Li, Haixun Wang, Kenny Q. Zhu, Zhongyuan Wang, and Xindong Wu. Computing term similarity by large probabilistic isa knowledge. In *CIKM*, pages 1401–1410, New York, NY, USA, 2013. ACM.

[MacKay, 2003] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[McCarthy *et al.*, 2007] Diana McCarthy, Sriram Venkatapathy, and Aravind K Joshi. Detecting compositionality of verb-object combinations using selectional preferences. In *EMNLP-CoNLL*, pages 369–379, 2007.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

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

[Mitchell and Lapata, 2010] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[Socher *et al.*, 2012] Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*, pages 1201–1211. ACL, 2012.

[Socher *et al.*, 2013a] Richard Socher, John Bauer, Christopher D Manning, et al. Parsing with compositional vector grammars. In *ACL*, pages 455–465, 2013.

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

[Tian *et al.*, 2017] Ran Tian, Naoaki Okazaki, and Kentaro Inui. The mechanism of additive composition. *Machine Learning*, 106(7):1083–1130, 2017.

[Venkatapathy and Joshi, 2005] Sriram Venkatapathy and Aravind K Joshi. Measuring the relative compositionality of verb-noun (vn) collocations by integrating features. In *HLT-EMNLP*, pages 899–906. ACL, 2005.

[Wang *et al.*, 2017] Shaonan Wang, Jiajun Zhang, and Chengqing Zong. Exploiting word internal structures for generic chinese sentence representation. In *EMNLP*, pages 298–303, 2017.

[Yazdani *et al.*, 2015] Majid Yazdani, Meghdad Farahmand, and James Henderson. Learning semantic composition to detect non-compositionality of multiword expressions. In *EMNLP*, pages 1733–1742, 2015.

[Yin *et al.*, 2016] Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. Unsupervised word and dependency path embeddings for aspect term extraction. *IJCAI*, pages 2979–2985, 2016.

[Yu and Dredze, 2015] Mo Yu and Mark Dredze. Learning composition models for phrase embeddings. *TACL*, 3:227–242, 2015.

[Zhang *et al.*, 2014] Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, Chengqing Zong, et al. Bilingually-constrained phrase embeddings for machine translation. In *ACL*, pages 111–121, 2014.