Phrase Type Sensitive Tensor Indexing Model for Semantic Composition

Yu Zhao[1], **Zhiyuan Liu**[1*], **Maosong Sun**[1,2]
[1] Department of Computer Science and Technology, State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing 100084, China
[2] Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu 221009, China
zhaoyu.thu@gmail.com, {liuzy, sms}@tsinghua.edu.cn

Abstract

Compositional semantic aims at constructing the meaning of phrases or sentences according to the compositionality of word meanings. In this paper, we propose to synchronously learn the representations of individual words and extracted high-frequency phrases. Representations of extracted phrases are considered as gold standard for constructing more general operations to compose the representation of unseen phrases. We propose a grammatical type specific model that improves the composition flexibility by adopting vector-tensor-vector operations. Our model embodies the compositional characteristics of traditional additive and multiplicative model. Empirical result shows that our model outperforms state-of-the-art composition methods in the task of computing phrase similarities.

Introduction

Compositional semantic aims at constructing the meaning of phrases or sentences according to the compositionality of word meanings. Most recently, continuous word representations are frequently used for representing the semantic meaning of words (Turney and Pantel 2010), which have achieved great success in various NLP tasks such as semantic role labeling (Collobert and Weston 2008), paraphrase detection (Socher et al. 2011a), sentiment analysis (Maas et al. 2011) and syntactic parsing (Socher et al. 2013a). Beyond word representation, it is also essential to find appropriate representations for phrases or longer utterances. Hence, compositional distributional semantic models (Marelli et al. 2014; Baroni and Zamparelli 2010; Grefenstette and Sadrzadeh 2011) have been proposed to construct the representations of phrases or sentences based on the representations of the words they contain.

Most existing compositional distributional semantic models can be divided into the following two typical types:

Vector-Vector Composition. These models use element-wise composition operations to compose word vectors into phrase vectors, as shown in Figure 1(A). For example, Mitchell and Lapata (2010) propose to use additive model ($\mathbf{z} = \mathbf{x} + \mathbf{y}$) and multiplicative model ($\mathbf{z} = \mathbf{x} \odot \mathbf{y}$). However, both of the operations are commutative, which may be unreasonable for semantic composition since the order of word sequences in a phrase may influence its meaning. For instance, *machine learning* and *learning machine* have different meanings, while the commutative functions will return the same representation for them.

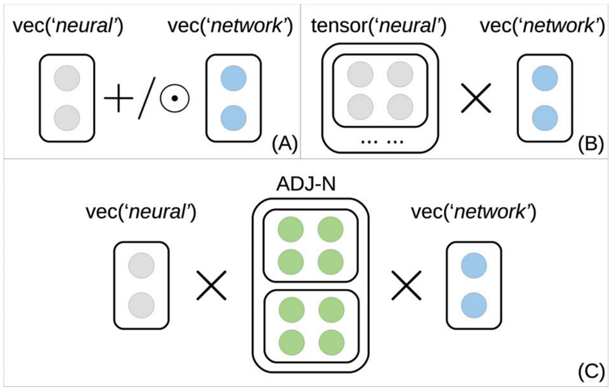


Figure 1: Comparison of different semantic composition models, including (a) vector-vector composition, (b) tensor-vector composition and (c) vector-tensor-vector composition.

Tensor-Vector Composition. To improve composition capability, complicated schemes for word representation are proposed to replace simple vector-space models, including matrices, tensors, or a combination of vectors and matrices (Erk and Padó 2008; Baroni and Zamparelli 2010; Yessenalina and Cardie 2011; Coecke, Sadrzadeh, and Clark 2010; Grefenstette et al. 2013). In this way, semantic composition is conducted via operations like tensor-vector product, as demonstrated in Figure 1(B) [1]. Despite of powerful capability, these methods have several disadvantages that reduce their scalability: (1) They have to learn matrices or tensors for each word, which is time-consuming. Moreover,

[*] Corresponding author: Zhiyuan Liu (liuzy@tsinghua.edu.cn).

[1] In accordant with grammatical structure, words with k arguments are represented by rank $k + 1$ tensors. Hence, the word *neural* is represented by a 2-order tensor, i.e., a matrix. An example for words represented by 3-order tensors is the verb *loves* in the clause *John loves Mary*.

$\mathbf{v}$(neural)	$\mathbf{v}$(network)	$\mathbf{v}$(neural) + $\mathbf{v}$(network)	$\mathbf{v}$(neural network)
neural	KAKEland	network	backpropagation
cortical	datacast	gene-regulatory	connectionist
synapses	cable	cellular	PCNN
sensory	IPTV	feedforward	self-organizing
axonal	broadcast	switched-circuit	probabilistic

Table 1: A comparison of the nearest neighbors of phrase *neural network* and its component words, measured by cosine similarity using compositional vector and learned phrase vector respectively.

there are insufficient data for low-frequency words. (2) They require rich linguistic knowledge such as word arguments, which are usually not available for all words in a large-scale corpus. (3) They are incapable of new words that have not appeared in training corpus, while this is a common phenomenon in real world.

An intuitive approach to construct phrase representation is to treat each phrase as a pseudo word to learn its representation by word representation models. The main idea is that representation of a phrase can also be learned in the same way as that of an individual word, inspired by the philosophy that the meaning of an expression is determined by how it is used (Clarke 2012). Take a two-word phrase *neural network* for example. Table 1 shows that the neighbor words of directly learned phrase vector are more revelent to *neural network* than those of compositional vector. However, it also suffers from data sparsity problem. Compared with words, there are more infrequent and out-of-vocabulary phrases, which are unavailable for representation learning.

Considering the above issues, we prefer to synchronously learn the representations of individual words and a collection of high-frequency phrases. The learned phrases representations are considered as gold standard so that the output representation of an ideal compositional function should be approximate to them. Such compositional function should be both non-commutative and non-lexicalized to avoid disadvantages of vector-vector and tensor-vector composition functions.

In this paper, we propose a vector-tensor-vector compositional function, named as Tensor Index Model (TIM). This composition function is composed of two matrices and a 3-order tensor, which projects two input word vectors to one phrase vector in the same semantic space, as shown in Figure 1(C). To enhance compositional capability, we distinguish various types of phrase structures, such as `Adj-N`, `N-N` and `V-Obj` [2]. A unique tensor-based composition function is trained for each phrase type. Our model can be easily extended to multi-word phrases or sentences via recursive binary operations.

In order to collect high-quality phrases and collocations to train composition functions, we use anchor texts from Wikipedia, which provide naturally annotated phrase boundaries. With the aid of state-of-the-art word representation models, we synchronously learn semantic representations of individual words and collected phrases. Parameters in composition function are estimated by minimizing the error between gold-standard phrase vectors and constructed compositional vectors. We evaluate our model on the phrase similarity computation task for `Adj-N`, `N-N` and `V-Obj` phrases respectively. Experimental results show that our model outperforms all the baselines reported in Blacoe and Lapata (2012).

Overview of Tensor Indexing Model

In this section, we present the framework of tensor indexing model (TIM), which generally captures compositional information for specific grammatical types. TIM integrates supervised parameter learning and unsupervised word representation together. Our framework for semantic composition can be divided into the following three steps.

Extracting Phrases. We use anchor texts in knowledge bases (i.e., Wikipedia in this paper) to collect high-quality phrases with various grammatical types. Occurrences of each phrase in the corpus are regarded as a pseudo word token, which will be used as the corpus for learning word and phrase representations. For example, the phrase *neural network* has appeared in the anchor texts of some Wikipedia links, hence we replace its occurrence in the corpus with a specific token *neural_network* in order to treat it as an individual word.

Constructing Representation. We synchronously learn word and phrase embeddings using the prepared corpus, projecting both words and phrases into the same semantic space. That is, vectors for the words like *neural* and *network* are learned together with vectors of phrases like *neural_network*.

Learning Composition Function. We build triples of phrase vectors and component word vectors as training samples for composition function, e.g. $(\mathbf{v}_{neural}, \mathbf{v}_{network}, \mathbf{v}_{neural\_network})$. We propose an effective supervised learning procedure to estimate parameters in the function.

Learning Tensor Composition

Tensor product, as a binary semantic composition, takes two d-dimensional word vectors $\mathbf{x}$ and $\mathbf{y}$ as input, and generates a d-dimensional phrase vector $\mathbf{z}$ as output. We denote f_r as the composition function corresponding to specific phrase type r such as `Adj-N`, consisting of both tensor product and linear transformation, which is formalized as:

$$\mathbf{z} = f_r(\mathbf{x}, \mathbf{y}) = W\mathbf{x}\mathbf{y} + M\mathbf{x} + N\mathbf{y}, \tag{1}$$

[2] Short for adjective-noun, noun-noun and verb-object phrases, respectively.

where $W \in \mathbb{R}^{d \times d \times d}$ is a 3-order tensor, and $M, N \in \mathbb{R}^{d \times d}$ are the linear transformation matrices. Each slice of the tensor acts as a coefficient matrix for one entry z_i in $\mathbf{z}$:

$$z_i = \sum_{j,k} W_{ijk} \cdot x_j \cdot y_k + \sum_{j} (M_{ij} \cdot x_j + N_{ij} \cdot y_j), \quad (2)$$

which can also rewritten as:

$$z_i = \mathbf{x}^T W_i \mathbf{y} + (M\mathbf{x})_i + (N\mathbf{y})_i. \quad (3)$$

It embodies the simple form of additive and multiplicative operations.

In practice, such tensor composition is infeasible due to high computational and memory complexity. To address this issue, we adopt low-rank tensor decomposition (Bai et al. 2009; Chen and Saad 2009). In this way, we approximate each slice of original tensor as the product of two low-rank matrices. For example, the ith slice of tensor W_i can be approximated as follows:

$$W_i \approx U_i^T V_i + I, \quad (4)$$

where U_i and V_i are $k \times d$ matrices and I is an identity matrix. In this paper, we restrict $k \leq 50$ considering the dimensionality of vector representation $d = 200$.

Based on the decomposition, the composition function for the ith component of $\mathbf{z}$ can approximated as:

$$z_i \approx (U_i\mathbf{x})^T (V_i\mathbf{y})^T + \mathbf{x}^T\mathbf{y} + (M\mathbf{x})_i + (N\mathbf{y})_i. \quad (5)$$

Due to the decomposition, the computation complexity drops from $\Theta(d^2)$ to $\Theta(kd)$ for each slice of tensor W. Moreover, the k-dimensional product of matrix-vector multiplication $U\mathbf{x}$, $V\mathbf{y}$, $M\mathbf{x}$ and $N\mathbf{y}$ can be pre-computed and stored in memory so that the complexity drops to a minimum of $\Theta(k + d)$ during the testing stage.

Suppose we are given a training set T_r of vector triples for each phrase type r. Each triple $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ contains semantic vectors of the words x and y and the phrase $z = xy$, in the same vector space. The training objective of TIM is to minimize the objective function on the training set:

$$J(\theta) = \sum_{t=1}^{|T_r|} \left(\mathbf{z}^{(t)} - f_r(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}, \theta) \right)^2 + \frac{\lambda}{2} \|\theta\|^2, \quad (6)$$

where θ is the set of all parameters W, M and N in TIM. The global error consists of both loss function and regularization term over all parameters, with λ as the regularization factor.

We train TIM using standard stochastic gradient descent (SGD). At each step, we randomly select a triple of ($\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$) from T_r and make the following updates:

$$\nabla\theta = \eta \cdot [\lambda\theta - 2 \cdot (\mathbf{z} - f_r(\mathbf{x}, \mathbf{y})) \cdot \frac{\partial f_r(\mathbf{x}, \mathbf{y})}{\partial \theta}], \quad (7)$$

where η is the learning rate. In our experiments, η is initialized to 10^{-4} and decreases 1% at the end of each iteration to avoid overfitting.

We compute derivatives for different parameters in θ respectively. Since we apply low-rank approximation on each slice of tensor W, the derivative should be adjusted to:

$$\frac{\partial f_i(\mathbf{x}, \mathbf{y})}{\partial U_i} = (V_i\mathbf{y}) \cdot \mathbf{x}^T, \quad \frac{\partial f_i(\mathbf{x}, \mathbf{y})}{\partial V_i} = (U_i\mathbf{x}) \cdot \mathbf{y}^T \quad (8)$$

In addition, beyond two-word phrases, it is easy to extend our composition function to represent vectors of multi-word expressions according to their grammatical structures. Given a text fragment as input, a phrase structure tree representation can be constructed using a standard constituency parser. Assuming a group of binary composition functions for all phrase categories are obtained, vectors of all non-terminal nodes can be recursively calculated with vectors of its children and the tensor corresponding to its grammar type.

Representation Construction

In this section we describe two strategies to collect naturally annotated phrases from Wikipedia. Different strategies will extract different numbers of phrases and generate dissimilar representations for the same phrase. Their performance will be compared in the experiments.

Phrase Extraction from Wikipedia

In the content of Wikipedia articles, there are hand-crafted anchor texts that connect to other related pages. We denote **anchor phrases** as the multi-word expressions labeled in anchor texts, which are regarded as naturally annotated collocations.

In this paper, we focus on extracting two-word anchor phrases for learning tensor composition functions. We employ the English Wikipedia corpus, which consists of 4,313,023 articles and 65,567,712 hyperlinks. In order to learn specific tensor composition functions for distinct phrase types, we employ part-of-speech (POS) tagging on Wikipedia articles with Stanford POS Tagger (Toutanova et al. 2003) to extract `Adj-N`, `N-N`, and `V-Obj` phrases from all candidates [3].

It is not surprising that most anchor phrases are nominal entities. Hence, we collect `V-Obj` phrases from the entire content of Wikipedia articles. Moreover, infrequent candidate phrases are insufficient for learning phrase representations, hence we only reserve the phrases occurring more than 20 times in the corpus for learning. Finally we obtain $93,183$ `Adj-N` phrases and $133,427$ `N-N` phrases and $14,506$ `V-Obj` phrases.

Corpus Construction

For each anchor phrase, we replace its occurrences in the corpus by a corresponding pseudo word. After the substitution operation, we obtain a text corpus for simultaneously learning word and phrase representations.

For `Adj-N` and `N-N` phrases, there are two combination strategies:

- Combining only those phrases inside anchor texts, denoted by AT (short for anchor texts).
- Combining all occurrences of anchor phrases in the corpus, denoted by FT (short for full texts).

We implement both strategies on the corpus to get two different representations for anchor phrases. According to the

[3] We select these three representative phrase types following the experimental setting in Mitchell and Lapata (2010)

w.r	c.m	Adj-N	N-N	V-Obj
SDS (BNC)	ADD	0.37	0.38	0.28
	MUL	0.48	0.50	0.35
	RAE	0.31	0.30	0.28
SDS (Wiki)	ADD	0.34	0.44	0.13
	MUL	0.42	0.63	0.28
	AT	0.42	0.45	-
	FT	0.49	0.57	-
SGM (Wiki)	ADD	0.73	0.73	0.62
	MUL	0.39	0.34	0.41
	AT	0.36	0.51	-
	FT	0.57	0.66	-
	TIM	**0.77**	**0.75**	**0.66**

Table 2: Spearman ρ correlation coefficients of composition models with human ratings, where w.r stands for word representation models, c.m stands for composition methods, AT and FT stands for learned phrase representations with different phrase extraction strategies.

statistics, the average frequency in FT of all obtained anchor phrases is $5,071$, only half of their occurrences are in anchor texts. In addition, 78% of anchor phrases occur more than 500 times in the corpus. The sufficiency of statistical data will ensure the rationality and reliability of phrase representation.

Representation Learning

There are various types of models to learn vector space representation of words. We mainly focus on the efficiency when it is used to train word vectors on large corpus like the whole English Wikipedia. For this task we employ Skip-gram Model (SGM), a log-linear architecture that can be efficiently trained on big data (Mikolov et al. 2013; Mikolov, Yih, and Zweig 2013).

SGM provides an approximate additive operation for measuring meaningful syntactic and semantic regularities. For example, vector("King") $-$ vector("Man") $+$ vector("Woman") results in a vector similar to vector("Queen"). Such interesting property suggests that SGM may contribute to capture additive compositionailty.

Experimental Results

In this paper, we evaluate our methodology by judging similarities between phrases. We first introduce the dataset used in the experiment and the state-of-the-art methods reported as the baselines. We compare additive and multiplicative composition functions with TIM model for different phrase types. We also analyze the influence of different parameters in detail to gain more insights.

Dataset and Evaluation Scheme

We use the phrasal similarity dataset described in Mitchell and Lapata (2010). This dataset contains 324 phrase pairs together with human judgements of pairwise similarity. Each sample in the dataset is a pair of phrases (p_1, p_2) and a human rating from 1 to 7, where a high number indicates higher similarity. We aggregate all the ratings for the same sample to obtain an overall judgment as the ground truth.

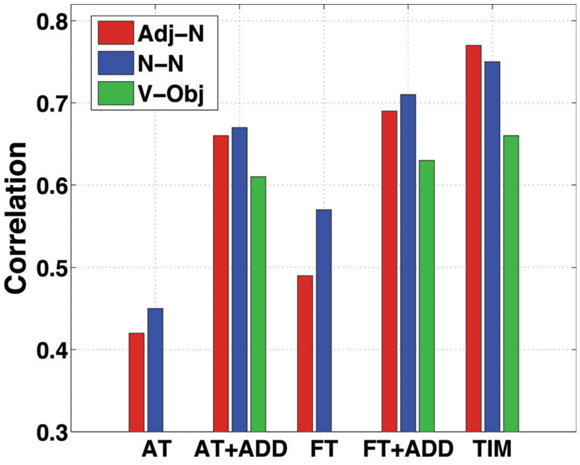


Figure 2: Results of combining learned phrase representations with basic composition model, using SGM for word representations.

Given a composition model C and a fixed vector representation for words, the phrase representation $V_C(p_1)$, $V_C(p_2)$ can be calculated for each sample, and the phrasal similarity between p_1 and p_2 can be computed via cosine similarity function:

$$Sim_C(p_1, p_2) = \frac{V_C(p_1) \cdot V_C(p_2)}{|V_C(p_1)| \cdot |V_C(p_2)|} \tag{9}$$

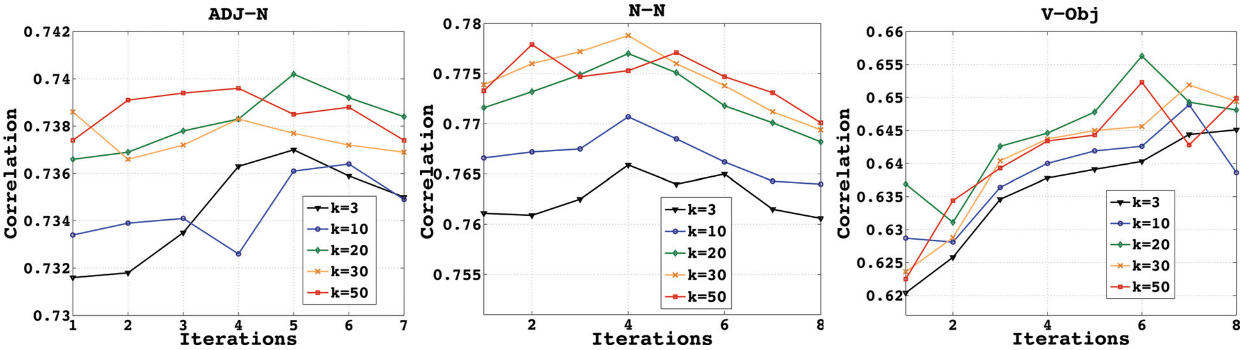
In our experiment, we evaluate traditional additive and multiplicative model using different word representations against the human similarity ratings using Spearman's ρ correlation coefficient.

Blacoe and Lapata (2012) made a comparison among three types of distributional representation for semantic composition. According to their report, a simple distributional semantic space (SDS) turned out to be the best word embedding for modeling phrase composition than the neural language model (Collobert and Weston 2008) and distributional memory tensor (Baroni and Lenci 2010). Thus, we choose SDS model as the baseline of modeling phrasal compositionality. Note that in the original paper, SDS model is trained on the British National Corpus (BNC), from where the sample phrases in dataset are selected. To maintain data consistency, we implement SDS model on the Wikipedia corpus, the performance of which is also considered as a baseline.

Result of Similarity Judgements

In this paper, we report results for all three phrase types in the dataset: adjective-noun (`Adj-N`), noun-noun (`N-N`) and verb-object (`V-Obj`). We consider both additive (ADD) and multiplicative (MUL) models. Result of recursive autoencoder (RAE) is also considered as baseline.

Table 2 demonstrates the performances of different composition methods. We find that Tensor Indexing Model per-



(a) influence of k for Adj-N phrases

(b) influence of k for N-N phrases

(c) influence of k for V-Obj phrases

Figure 3: Influences of dimension k for tensor decomposition in TIM for predicting phrase similarity.

forms the best across all phrase categories. TIM achieves an average of 0.73 on Spearman ρ correlation coefficient with human ratings, which is at least 0.03 higher than any other baseline. This result shows that our model successfully captures the compositionality between component words in phrases. We also notice that traditional additive function under SGM achieves a fairly good performance. Compared with the baseline SDS model, it achieves an average of 0.25 improvement on Spearman ρ correlation coefficient.

In addition, additive model is obviously more effective than multiplicative model using SGM for word representations, which is contrary to SDS model. It can be inferred that SGM captures the additive compositionality between individual words. In the experiment settings, we initialize tensors for all phrase categories to be empty with minor random noises, but they turn out to be extremely dense in each slice after the training stage. Since the traditional multiplicative function is a simple special case of TIM with a diagonal identity tensor, it can hardly obtain comparable performance.

As described in the previous section, we employ two strategies to learn representations for anchor phrases. In Table 2 we find that FT strategy is significantly better than AT strategy, which indicates that the sufficiency of statistical data has a great impact on the quality of phrase embedding. Despite of the limited number of phrases discovered, FT strategy outperforms SDS model, which suggests that it is effective to learn phrase representations directly.

The reason why FT strategy cannot be directly used to model semantic composition is that anchor phrases only cover a minority of all possible phrases. There are only 28 adjective-noun phrases, 43 noun-noun phrases and none verb-noun phrases appearing in both Wikipedia and test dataset. To avoid the sparsity problem, we compare different combinations of phrase embedding strategies and composition functions. Figure 2 shows that the FT strategy with additive model (FT + ADD) works well under SGM word representations. It achieves 0.69, 0.71 and 0.63 on Spearman ρ correlation coefficient for three phrase types respectively, which is comparable to the performance of TIM. We notice that the result for `Adj-N` phrases of TIM is better than the result for `N-N` phrases. It indicates that multiplicative function also contribute to TIM, since multiplicative model perform better for `N-N` phrases (0.39) than for `Adj-N` phrases (0.34) using SGM for word representations.

Influence of Tensor Decomposition

We analyze the influence of dimension k for tensor decomposition. As we can see from Figure 3(a) to 3(c), the choice of k affects the general performance more significantly than other hyper-parameters. We make two observations: (1) The performance improves when k increases. (2) The benefit of choosing larger k drops off when it gets to higher than 30.

The explanation is that larger size of k introduce more variables in the approximated tensor which introduces the problem of data sparsity. Although it leads to more powerful capability, the model cannot be fully trained with insufficient training phrases [4]. Moreover, time complexity is proportional to the tensor size. It is inappropriate to greedily raise the value of k.

Visualization and Case Study

To demonstrate the effectiveness of semantic composition, we present a visualization of high-dimensional phrase representations [5], as shown in Figure 4. We select three pairs of semantically related phrases, and for each of them collect a cluster of 30 most similar phrases. The visualization indicates that semantically related phrases are closer together, such as *machine learning* and *neural network*, *telephone number* and *phone call*, or *black hair* and *blue eye*. Furthermore, there are overlaps between those clusters of similar phrases, which means that similar phrases share some of the same neighbors.

In order to gain more information on the strength and limitation of our framework, we provide examples of severe conflictions between TIM and human ratings, as can be seen in Table 3. We find that TIM considers both synonym and antonym relations as relevant and assigns high

[4] We obtain more training phrases for N-N than for V-Obj, so that the best fitting k for N-N type is higher.

[5] We use the t-SNE toolkit for visualization. https://github.com/turian/textSNE

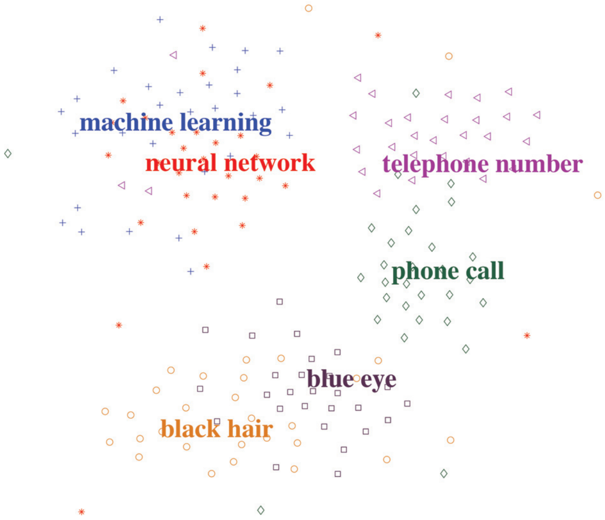


Figure 4: Examples of 2-d visualization of high-dimensional phrase representations.

similarity scores (e.g, *cold air* and *hot weather*), while human judgments tend to assign higher scores only to synonym relations. The reason is that both synonyms and antonyms tend to share similar context environment, as TIM constructs representations for training phrases based on contextual information.

We also present several examples in which TIM performs badly. TIM is not capable of identifying implicit relations which is usually straight forward to humans. For instance, the correlation between *meet requirement* and *win battle* originates in human's feeling and spiritual experience. They are considered very related by humans, while it is hard to recognize from concurrences of specific vocabulary in textual corpus.

Another limitation of TIM is the lack of prior world knowledge. For example, *health minister* and *town hall* relate to a common topic so that humans tend to find them similar. It is hard to infer such relation by a general composition function for all N-N phrases, since the contextual topic distribution of component words and that of the entire phrase are not the same.

Related Work

Besides simple operations such as vector addition and element-wise multiplication (Mitchell and Lapata 2010), more complicated functions (Clark and Pulman 2007; Clark, Coecke, and Sadrzadeh 2008; Coecke, Sadrzadeh, and Clark 2010) have been proposed for semantic composition, which usually represent words in semantic spaces of different orders according to their grammatical types. For example, the words indicating relational types are represented by matrices and argument words are represented by vectors (Grefenstette and Sadrzadeh 2011). Grefenstette (2013) further extend the representation of verbs using high order tensors, with multi-step regression for parameter learning. In accordant with grammatical structure, words with k arguments are represented by rank $k+1$ tensors.

Phrase 1	Phrase 2	TIM Rank	Human Rank
high price	low cost	2	60
cold air	hot weather	7	56
health minister	office worker	40	81
increase number	reduce amount	4	67
receive letter	send message	9	71
cause injury	suffer loss	7	68
early age	new life	79	17
good place	high point	61	16
county council	town hall	59	14
defence minister	security policy	48	4
meet requirement	win battle	94	33
express view	share interest	60	7

Table 3: Most severe conflictions between human judgments and TIM decisions in rating phrase similarity, ranging from rank 1 to rank 108.

Moreover, Yessenalina and Cardie (2011) represented each word as a separate matrix, and modeled the semantic composition between neighboring word pairs as iterated matrix multiplication. Socher et al. (2012) associated each word with a vector-matrix pair, and introduced a recursive matrix-vector neural network for semantic composition operation, which can be regarded as being generalized from linear models (Zanzotto et al. 2010) and matrix operations (Socher et al. 2011b). Baroni and Zamparelli (2010) constructed the vectors of adjective-noun pairs by multiplying matrices of adjectives with vectors of argument nouns.

The above mentioned methods, to some extent, aim at improving representation capability of composition functions by increasing representation complexity of individual words. However, as we have stated in the introduction, they will suffer from the sparsity problem and is time-consuming for learning on large-scale corpora.

The work most similar to us is Socher et al. (2013b), which adopted tensors to capture relations between entities in knowledge bases aiming at relation prediction. They also proposed a recursive neural tensor network towards understanding compositionality for sentiment analysis (Socher et al. 2013c). The main difference is that they did not construct the representation of interior nodes beforehand and their training objective depends on specific task, while we adopt tensor as a general operation for semantic composition.

Conclusions

In this paper, we introduce a tensor based composition function for different phrase types. We synchronously learn word and phrase representations in the same semantic vector space. We propose to extract phrases from anchor texts in Wikipedia, which can be regarded as naturally annotated phrase boundaries. Experiment result shows that our model achieves an excellent performance in the task of phrase similarity judgment, in comparisons of traditional vector composition models.

Acknowledgments

This research is supported by the 973 Program (No. 2014CB340501) and the National Natural Science Foundation of China (NSFC No. 61133012 & 61170196).

References

Bai, B.; Weston, J.; Grangier, D.; Collobert, R.; Sadamasa, K.; Qi, Y.; Cortes, C.; and Mohri, M. 2009. Polynomial semantic indexing. In *NIPS*, 64–72.

Baroni, M., and Lenci, A. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics* 36(4):673–721.

Baroni, M., and Zamparelli, R. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, 1183–1193. Association for Computational Linguistics.

Blacoe, W., and Lapata, M. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, 546–556. Association for Computational Linguistics.

Chen, J., and Saad, Y. 2009. On the tensor svd and the optimal low rank orthogonal approximation of tensors. *SIAM Journal on Matrix Analysis and Applications* 30(4):1709–1734.

Clark, S., and Pulman, S. 2007. Combining symbolic and distributional models of meaning. In *AAAI Spring Symposium: Quantum Interaction*, 52–55.

Clark, S.; Coecke, B.; and Sadrzadeh, M. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, 133–140.

Clarke, D. 2012. A context-theoretic framework for compositionality in distributional semantics. *Computational Linguistics* 38(1):41–71.

Coecke, B.; Sadrzadeh, M.; and Clark, S. 2010. Mathematical foundations for a compositional distributional model of meaning. *CoRR* abs/1003.4394.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International conference on Machine Learning*, 160–167. ACM.

Erk, K., and Padó, S. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, 897–906. Association for Computational Linguistics.

Grefenstette, E., and Sadrzadeh, M. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, 1394–1404. Association for Computational Linguistics.

Grefenstette, E.; Dinu, G.; Zhang, Y.-Z.; Sadrzadeh, M.; and Baroni, M. 2013. Multi-step regression learning for compositional distributional semantics. *IWCS*.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 142–150. Association for Computational Linguistics.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Yih, W.-t.; and Zweig, G. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, 746–751.

Mitchell, J., and Lapata, M. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Socher, R.; Huang, E. H.; Pennington, J.; Ng, A. Y.; and Manning, C. D. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, volume 24, 801–809.

Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; and Manning, C. D. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, 151–161. Association for Computational Linguistics.

Socher, R.; Huval, B.; Manning, C. D.; and Ng, A. Y. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, 1201–1211.

Socher, R.; Bauer, J.; Manning, C. D.; and Ng, A. Y. 2013a. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013b. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of NIPS*, 926–934.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013c. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 1631–1642.

Toutanova, K.; Klein, D.; Manning, C. D.; and Singer, Y. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of ACL-HLT*, 173–180.

Turney, P., and Pantel, P. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37(1):141–188.

Yessenalina, A., and Cardie, C. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of EMNLP*, 172–182.

Zanzotto, F. M.; Korkontzelos, I.; Fallucchi, F.; and Manandhar, S. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, 1263–1271.