Zhu JZ, Jia YT, Xu J *et al.* Modeling the correlations of relations for knowledge graph embedding. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 33(2): 323–334 Mar. 2018. DOI 10.1007/s11390-018-1821-8

Modeling the Correlations of Relations for Knowledge Graph Embedding

Ji-Zhao Zhu[1,2], Yan-Tao Jia[2], *Member*, *CCF*, *ACM*, Jun Xu[2,*], *Member*, *CCF*, *ACM*, *IEEE*
Jian-Zhong Qiao[1,*], *Senior Member*, *CCF*, and Xue-Qi Cheng[2], *Fellow*, *CCF*, *Member*, *ACM*, *IEEE*

[1] *College of Computer Science and Engineering, Northeastern University, Shenyang 110169, China*
[2] *Key Laboratory of Network Data Science and Technology, Institute of Computing Technology*
Chinese Academy of Sciences, Beijing 110190, China

E-mail: zhujz.neu@gmail.com; {jiayantao, junxu}@ict.ac.cn; qiaojianzhong@mail.neu.edu.cn; cxq@ict.ac.cn

Received January 20, 2017; revised August 24, 2017.

Abstract Knowledge graph embedding, which maps the entities and relations into low-dimensional vector spaces, has demonstrated its effectiveness in many tasks such as link prediction and relation extraction. Typical methods include TransE, TransH, and TransR. All these methods map different relations into the vector space separately and the intrinsic correlations of these relations are ignored. It is obvious that there exist some correlations among relations because different relations may connect to a common entity. For example, the triples (Steve Jobs, PlaceOfBrith, California) and (Apple Inc., Location, California) share the same entity California as their tail entity. We analyze the embedded relation matrices learned by TransE/TransH/TransR, and find that the correlations of relations do exist and they are showed as low-rank structure over the embedded relation matrix. It is natural to ask whether we can leverage these correlations to learn better embeddings for the entities and relations in a knowledge graph. In this paper, we propose to learn the embedded relation matrix by decomposing it as a product of two low-dimensional matrices, for characterizing the low-rank structure. The proposed method, called TransCoRe (Translation-Based Method via Modeling the Correlations of Relations), learns the embeddings of entities and relations with translation-based framework. Experimental results based on the benchmark datasets of WordNet and Freebase demonstrate that our method outperforms the typical baselines on link prediction and triple classification tasks.

Keywords knowledge graph embedding, low-rank, matrix decomposition

1 Introduction

A knowledge graph represents the entities and relations with nodes and edges, respectively. Popular knowledge graphs include WordNet[1], Freebase[2-3], YAGO[4], etc., and they are now freely available on the Internet. A great number of applications have been advanced by using these knowledge graphs, such as knowledge graph completion[5-6], document understanding[7], and digital assistants. However, knowledge graphs are commonly constructed and validated by human experts' efforts based on different rigorous symbols and they usually contain millions of vertices and billions of edges modeled as triples of the form (head entity, relation, tail entity) (denoted as (h, r, t)). It is difficult and inefficient to handle the applications over the huge amount of elements for any inference, query and computation. For example, when using Freebase for link prediction, we need to deal with 68 million of vertices and one billion of edges. Meanwhile, to query a simple entity or relation over the huge amount of elements in a knowledge graph needs more understandable and effective query language settings and techniques than that in a knowledge graph with a small size[8].

Regular Paper

This work was supported by the National Basic Research 973 Program of China under Grant No. 2014CB340405, the National Key Research and Development Program of China under Grant No. 2016YFB1000902, and the National Natural Science Foundation of China under Grant Nos. 61402442, 61272177, 61173008, 61232010, 61303244, 61572469, 91646120 and 61572473.

*Corresponding Author

In recent years, knowledge graph embedding technique has been proposed for representing the entities and relations as latent numerical vectors and hiding the underlying differences of knowledge graphs, while it can preserve the intrinsic structures of the original graph. The typical knowledge graph embedding methods include TransE[9], TransH[10], TransR[11], and so on.

In spite of their success in modeling knowledge graphs, these methods map different relations into the vector space separately and the intrinsic correlations among these different relations are ignored. However, it seems that there exist correlations among the different relations, because it is a common phenomenon that different relations connect to a common entity. For example, the triples (Steve Jobs, PlaceOfBrith, California) and (Apple Inc., Location, California) share the same entity California as their tail entity. Statistics on the dataset of FB15K indicate that 99.13% entities are directly linked by two or more different relations.

Further analysis based on the embedded relation matrix, consisting of relation vectors by column, indicates that the correlations do exist and can be modeled as a low-rank structure upon the embedded relation matrix. To be specific, we first learn an embedded relation matrix with the existing method of TransE, based on the dataset of FB15K. Then, we conduct singular value decomposition (SVD)[12-13] on the matrix. The result shows that 80% of the energy can be captured by only about 20%∼30% of the top dimensions. Similar phenomena can also be observed from the matrices learned by other embedding methods of TransH and TransR. The results reveal that there exist correlations among the relations and the correlations are showed as a low-rank structure upon the embedded relation matrix.

It is natural to ask whether we can leverage the correlations to learn better embeddings for the entities and relations in a knowledge graph. In this paper, we propose a novel knowledge graph embedding method, called TransCoRe (Translation-Based Method via Modeling the Correlations of Relations), to learn the relation embeddings by decomposing the embedded relation matrix into two low-dimensional matrices for characterizing the low-rank structure explicitly. In this way, the problem of learning the embedded relation matrix is converted into learning the two low-dimensional matrices, and the correlations of relations will be captured during training. With the proposed method, we conduct experiments on the benchmark datasets of WordNet and Freebase, and the results show that our new method outperforms the typical methods including TransE, TransH, and TransR on the tasks of link prediction and triple classification.

2 Related Work

For the past few years, knowledge graph embedding technique has received considerable attention, as it can hide the underlying differences of knowledge graphs by representing both the entities and relations as low-dimensional vectors in a continuous vector space. A great many methods for knowledge graph embedding have been proposed by using this technique and they can be roughly divided into two major categories, translation-based and tensor decomposition based methods.

2.1 Translation-Based Methods

Inspired by the framework introduced in [14], TransE[9] represents relations as translations in the embedding space and assumes that $\boldsymbol{h}+\boldsymbol{r}=\boldsymbol{t}$, if (h,r,t) is a golden triple (please note that we use the same letter in boldface to represent their embedding vector). This indicates that $\boldsymbol{t}$ should be a nearest neighbor of $\boldsymbol{h}+\boldsymbol{r}$, whereas $\boldsymbol{t}$ should be far away from $\boldsymbol{h}+\boldsymbol{r}$. TransE is efficient and suitable well for 1-to-1 relations, but it has problems for n-to-1, 1-to-n and m-to-n relations. To overcome these problems, TransH[10] models a relation as a hyperplane together with a translation operation on it. Each relation is represented by two vectors. One is the norm vector $\boldsymbol{w}_r$ of the hyperplane, and the other is the translation vector $\boldsymbol{d}_r$ on the hyperplane. The embeddings for head entity $\boldsymbol{h}$ and tail entity $\boldsymbol{t}$ are projected to the hyperplan of relation r, and the projected vectors can be obtained $\boldsymbol{h}_\perp=\boldsymbol{h}-\boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{h}\boldsymbol{w}_r$ and $\boldsymbol{t}_\perp=\boldsymbol{t}-\boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{t}\boldsymbol{w}_r$. Both TransE and TransH embed the entities and the relations in the same vector space. In fact, an entity may have different aspects, and various relations focus on different aspects of entities. Hence, it is intuitive that some entities should be close to one another in a semantic space if they are similar, while they may be far away in other different semantic spaces. To model entities and relations in separate spaces, TransR[11] is proposed to model entities and relations in distinct spaces (i.e., entity space and multiple relation spaces) and perform translation in the corresponding relation space. In TransR, for each relation r, a projection matrix $\boldsymbol{M}_r$, and an embedding vector $\boldsymbol{r}$ are set for it. For each triple (h,r,t), entities in the entity space are first projected into r-relation space to get $\boldsymbol{h}_r=\boldsymbol{h}\boldsymbol{M}_r$ and $\boldsymbol{t}_r=\boldsymbol{t}\boldsymbol{M}_r$.

2.2 Tensor Decomposition Based Methods

The category of tensor decomposition based embedding methods are a well developed mathematical tool for data analysis. Some notable studies belonging to this category proposed in recently years, include RESCAL[15], TripleRank[16], TRESCAL[17], and TUCKER[18]. RESCAL is a typical tensor decomposition based method. Compared with other tensor decomposition methods, the main advantage of RESCAL is that it can exploit a collective learning effect when applied to relational data. RESCAL has shown very good results in various canonical relational learning tasks such as link predication.

TripleRank is similar to RESCAL, and it applies the CP (CANDECOMP/PARAFAC)[19] tensor decomposition to resource description framework (RDF) graphs for faceted browsing. However, in contrast to the tensor decomposition employed in RESCAL, CP is not capable of collective learning. TRESCAL can be viewed as an extension of RESCAL, which tries to encode rules into RESCAL. However, it focuses solely on a single rule, i.e., the arguments of a relation should be entities of certain types. TUCKER decomposition, also known as high-order singular value decomposition, factorizes a tensor into a core tensor multiplied by a matrix along each dimension.

In addition to the above methods, there are also many other energy-based methods that assign low energy to the plausible triples existing in a knowledge graph, employing neural network for learning. For instance, unstructured method[20-21] can be viewed as a naive version of TransE without considering the differences of relations, leading to score function $f_r(h,t) = \|\boldsymbol{h}-\boldsymbol{t}\|_2^2$. Structured embedding (SE)[22] defines a pair of relation-specific matrices ($\boldsymbol{M}_{rh}$, $\boldsymbol{M}_{rt}$) for head and tail entity, and defines the score function as an L_1 distance between the two projected vectors, namely, $f_r(h,t) = \|\boldsymbol{M}_{rh}\boldsymbol{h} - \boldsymbol{M}_{rt}\boldsymbol{t}\|_{l_1}$. Because SE has two separate matrices for optimization, it cannot capture the precise relationships between entities and relations. Single Matching Energy (SME) model[21] aims to capture the correlations between entities and relations via multiple matrix products and Hadamard product. SME simply represents each relation using a single vector, which interacts with entity vectors via linear matrix products, with all relations sharing the same parameters. Latent Factor Model (LFM)[23] encodes the entity as a vector and assigns a matrix for each relation. LFM incorporates the interaction of the two entity vectors in a simple and effective way by defining a bilinear score function $f_r(h,t) = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{M}_r\boldsymbol{t}$. Single Layer Model (SLM) is a naive baseline of NTN[24] by concatenating h and t as an input layer to a non-linear hidden layer. Neural Tensor Network (NTN) extends SLM by considering the second-order correlations in nonlinear neural networks and defines an expressive score function as $f_r(h,t) = \boldsymbol{u}_r^{\mathrm{T}} f(\boldsymbol{h}^{\mathrm{T}}\boldsymbol{M}_r\boldsymbol{t} + \boldsymbol{M}_{r1}\boldsymbol{h} + \boldsymbol{M}_{r2}\boldsymbol{t} + \boldsymbol{b}_r)$. However, the corresponding high complexity of NTN may prevent it from efficiently applying on large-scale knowledge graphs.

Despite the effectiveness of these aforementioned methods, they ignore the intrinsic relationships of relations and learn the vector representations for relations separately. In this paper, we test the hypothesis that better representations can be obtained by capturing the correlations of relations during training.

3 Analysis on Correlations of Relations

In this section, we firstly propose to analyze the correlations among relations via applying SVD to a learned embedded relation matrix. Specifically, given an embedded relation matrix $\boldsymbol{R} \in \mathbb{R}^{d\times N_r}$ learned by a state-of-the-art method, where N_r is the number of relations and d is the dimension of embedded space. We perform SVD on matrix $\boldsymbol{R}$ and sort the eigenvalues in descending order. We calculate the percentages of the energy being captured when different numbers of top eigenvalues (dimensions) are kept. Fig.1 illustrates the percentages of energy w.r.t. the number of dimensions. In the analysis, three embedded relation matrices are learned based on FB15K (containing $N_r = 1\,345$ relations) by the methods of TransE, TransH, and TransR, respectively. The embedded dimension d is set to 200 when learning these matrices (please note that $d < N_r$).

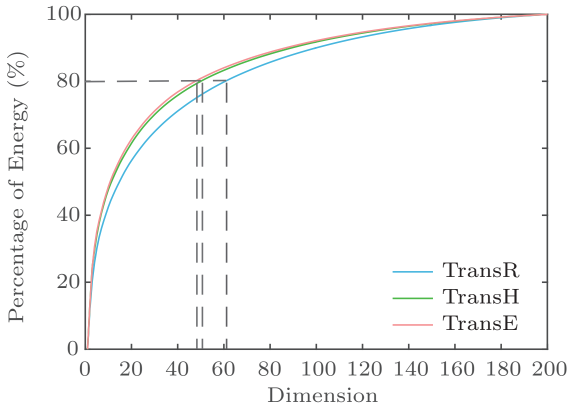


Fig.1. Percentages of energy w.r.t. the number of kept dimensions.

From the results shown in Fig.1, we can see that for the embedded relation matrix learned by TransE, we can capture 80% of the energy with only 46 (23%) top dimensions (the red curve in Fig.1). If we want to capture 100% of the energy, we need 100% of the dimensions. The results indicate that there exists a low-rank structure over the learned embedded relation matrix, which may result from the correlations of relations.

Similar phenomena can also be observed from the embedded relation matrices learned by other methods. For example, based on the learned embedded relations matrix by TransH and TransR, we only need 25% and 30% of the top dimensions if we want to capture 80% of the energy in the matrices (green curve and blue curve, respectively).

In other ways, we also take the Pearson correlation coefficient (PCC)[25] to demonstrate the existence of correlations among relations. Firstly, we calculate the PCCs between each relation pair based on the embedded relation matrix $\boldsymbol{R}$, and obtain a symmetric matrix of PCCs denoted as $\boldsymbol{P}$, where the values of the i-th row (or column) vector represent the degree of the correlation between the i-th relation and each of the relation set, and the diagonal values are always 1.0. Afterwards, we calculate the percentages of correlated relations above the given threshold of PCC. Concretely, for the i-th relation, if any value of the absolute entities of the i-th row (or column) vector, excepting the i-th entity, is greater than or equal to the given threshold, the number of the correlated relations will be increased by 1. Finally, we obtain the results illustrated in Fig.2. From the results we can see that: 1) more than 98% relations are weakly correlated (i.e., $PCC \geqslant 0.2$); 2) about 71% relations have medium correlations (i.e., $PCC \geqslant 0.4$); 3) there are 40% relations with strong correlations (i.e., $PCC \geqslant 0.6$); 4) almost 13% relations have very strong correlations (i.e., $PCC \geqslant 0.8$).

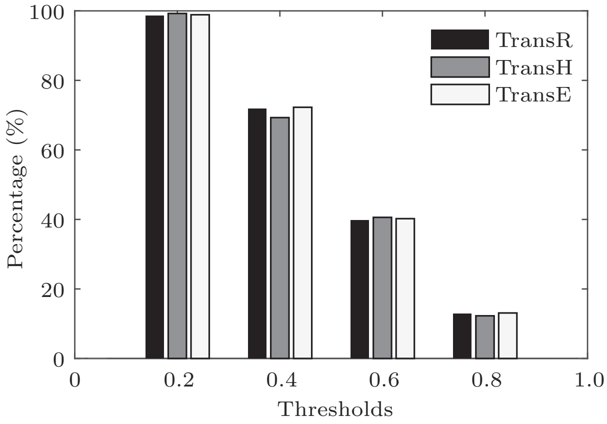


Fig.2. Percentages of correlated relations above different thresholds of Pearson correlation coefficient (PCC). PCC which lies in the intervals $[0, 0.2)$, $[0.2, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.8)$, $[0.8, 1.0]$ means the degrees of correlation are very weak, weak, medium, strong and very strong, respectively.

4 Our Method

Based on the analysis in Section 3, we introduce our method to characterize the correlations of relations. Let us firstly introduce some notations. We use the triples of form (h, r, t), where h and t denote the head entity and the tail entity respectively, and r denotes the relation from h to t, to model the facts in a knowledge graph. The sets of entity (including head and tail entities) and relation are denoted by $\mathcal{E}$ and $\mathcal{R}$ respectively, and N_e, N_r represent the size of $\mathcal{E}$ and $\mathcal{R}$, respectively. Δ represents the set of positive triples existing in the knowledge graph, and Δ' represents the set of negative triples corrupted from positive ones. Let $\boldsymbol{R} = (\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_{N_r})$ denote the embedded relation matrix, where $\boldsymbol{r}_i$ stands for the embedding vector of the i-th relation r.

4.1 Modeling the Correlations of Relations

As analyzed in Section 3, the correlations of relations are shown as a low-rank structure over the embedded relation matrix. We propose to learn the relation matrix $\boldsymbol{R}$ by decomposing it as a product of two low-dimensional matrices. Let $\boldsymbol{U} \in \mathbb{R}^{d \times k}$ be the left matrix consisting of column vectors $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_i, \ldots, \boldsymbol{u}_k)$ and $\boldsymbol{V} \in \mathbb{R}^{k \times N_r}$ be the right matrix consisting of column vectors $(\boldsymbol{v}_1, \ldots, \boldsymbol{v}_j, \ldots, \boldsymbol{v}_{N_r})$, where the dimension of column vector $\boldsymbol{u}_i$ is d, that of $\boldsymbol{v}_j$ is k, and k $(0 < k \leqslant \min(d, N_r))$ is a tunable parameter that represents the rank of $\boldsymbol{R}$. We get that

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}.$$

As illustrated in Fig.3, each column vector $\boldsymbol{r}$ of $\boldsymbol{R}$ can be written as a linear combination of the column vectors of matrix $\boldsymbol{U}$ multiplied by the corresponding vector $\boldsymbol{v}$ of coefficient matrix $\boldsymbol{V}$. Namely, for each column vectors $\boldsymbol{r}_i$ in $\boldsymbol{R}$, we have

$$\begin{aligned}\boldsymbol{r}_1 &= \boldsymbol{U}\boldsymbol{v}_1,\\ \boldsymbol{r}_2 &= \boldsymbol{U}\boldsymbol{v}_2,\\ &\vdots\\ \boldsymbol{r}_{N_r} &= \boldsymbol{U}\boldsymbol{v}_{N_r}.\end{aligned}$$

That is to say, the column vectors of $\boldsymbol{U}$ are the basis of the relation space. Therefore, the problem of learning relation embeddings is converted into that of learning the two low-dimensional matrices. In this way, the embedding vectors of all relations share the same matrix $\boldsymbol{U}$, and different relations have different v_i for $i = 1, 2, \ldots, N_r$. In this sense, the matrix $\boldsymbol{U}$ captures the common information of all relations.

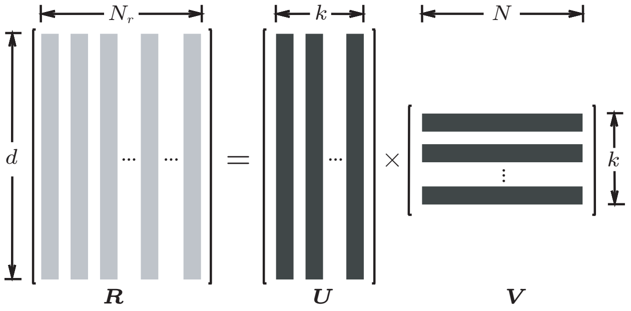


Fig.3. Illustration of the embedded relation matrix decomposed as a product of two matrices.

4.2 Translation-Based Method TransCoRe

Following the practices in [9], we adopt the translation-based method for learning the embeddings of entities and relations. Specifically, to determine the embeddings of entities, the element values in matrix $\boldsymbol{U}$ and vector $\boldsymbol{v}$, we construct the score function $f_r(h, t)$ for each triple (h, r, t):

$$f_r(h,t) = \|\boldsymbol{h} + \boldsymbol{U}\boldsymbol{v} - \boldsymbol{t}\|_2^2, \tag{1}$$

where $\boldsymbol{h}, \boldsymbol{t}$ are the embeddings for the head and tail entities, respectively. Intuitively, (1) implies that $\boldsymbol{U}\boldsymbol{v}$ is a translation from $\boldsymbol{h}$ to $\boldsymbol{t}$ since $f_r(h,t) = 0$, when $\boldsymbol{h} + \boldsymbol{U}\boldsymbol{v} = \boldsymbol{t}$.

To further determine the parameters in $f(h, r, t)$, we minimize the following margin-based ranking loss function

$$\mathcal{L} = \sum_{(h,r,t)\in\Delta} \sum_{(h',r,t')\in\Delta'} [f_r(h,t) + \gamma - f_r(h',t')]_+, \tag{2}$$

where $[\cdot]_+ = \max(0, \cdot)$ is the hinge function, γ is the margin separating the positive triples from the negative ones, and

$$\Delta' = \{(h', r, t)|h' \in \mathcal{E}\} \cup \{(h, r, t')|t' \in \mathcal{E}\},$$

is the set of random sampled negative triples. In practice, the set of negative triples Δ' is constructed from correct triple (h, r, t) by replacing the head entity h and/or the tail entity t with h' and/or t'. We follow the practice in [10] and set different probabilities for replacing the head and the tail entity depending on the mapping property of the relation. In our experiments, we adopt both of the popularly used strategies of "unif" and "bern" to construct negative labels. "unif" is the traditional sampling method that replaces h or t with identical probabilities to obtain a corrupted triple, and "bern" is the new method proposed in [10] which replaces h or t with different probabilities. Additionally, we enforce the following constraints in our training: $\|\boldsymbol{h}\|_2 \leqslant 1$, $\|\boldsymbol{t}\|_2 \leqslant 1$, $\|\boldsymbol{u}\|_2 = 1$, where $\boldsymbol{u}$ is the column vector of $\boldsymbol{U}$.

In our experiments, all embeddings for entities and relations are randomly initialized from $U(-\frac{6}{\sqrt{d}}, \frac{6}{\sqrt{d}})$ as in [9] and jointly learned via minimizing the loss function (2) during training. Moreover, the stochastic gradient descent (SGD) method is used during the optimization process.

4.3 Complexity Analysis

The space complexity of TrnasCoRe is $O(N_e m + N_r k + nk)$, where m is the dimension of entity embedding, n is the dimension of relation embedding, $N_e m$ is used for storing the embeddings of entities, $N_r k$ is used for storing the linearly independent vectors, and nk is used for storing the coefficient matrix.

We simply suppose the time complexity of operation $\|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|$ is a unit as in [26]. For our method, to get each element of the relation embedding vector r_{ij} needs k operations. Therefore, the time complexity is

$$T(N_t) = N_t + kN_t = O((1 + k)N_t),$$

where N_t is the number of triples in a knowledge graph.

Table 1 lists the complexities of several methods described in related work. In practice, parameter k is not greater than $nN_r/(n + N_r)$ for k, n, N_r are positive integers and $N_r > 2$, and thus our method has the lowest space complexity. From the last column of Table 1, we could see that the time complexity of our method is slightly higher than those of TransE and Unstructured which are $O(N_t)$, but lower than those of other baselines.

5 Experiments and Analysis

In order to evaluate against existing methods, we conduct experiments on two tasks: link prediction[22] and triple classification[9-11]. Next, we show the experimental results and their analysis.

Table 1. Complexity Analysis

Method	Space Complexity	Time Complexity
Unstructured	$O(N_em)$	$O(N_t)$
SE	$O(N_em + 2N_rn^2)(m = n)$	$O(2m^2N_t)$
SME(linear)	$O(N_em + N_rn + 4ml + 4l)(m = n)$	$O(4mlN_t)$
SME(bilinear)	$O(N_em + N_rn + 4mls + 4l)(m = n)$	$O(4mlsN_t)$
LFM	$O(N_em + N_rn^2)(m = n)$	$O(m(1 + m)N_t)$
SLM	$O(N_em + N_r(2l + 2ln))(m = n)$	$O(l(2m + 1)N_t)$
NTN	$O(N_em + N_r(n^2s + 2ns + 2s))(m = n)$	$O((l(2m + 1) + ms(1 + m))N_t)$
TransE	$O(N_em + N_rn)(m = n)$	$O(N_t)$
TransH	$O(N_em + 2N_rn)(m = n)$	$O(2mN_t)$
TransR	$O(N_em + N_r(1 + m)n)$	$O(2mnN_t)$
Proposed TransCoRe	$O(N_em + N_rk + nk)(m = n)$	$O((1 + k)N_t)$

Note: N_t denotes the number of triples in a knowledge graph. m is the dimension of the entity embedding vector and n is the dimension of the relation embedding vector. l is the number of hidden nodes of a neural network and s is the slice number of a tensor.

5.1 Datasets

The link prediction and triple classification tasks are implemented on two typical knowledge graphs: WordNet[1] and Freebase[3]. WordNet is a large lexical knowledge graph which provides semantic knowledge of words. In this paper, we employ two subsets from WordNet: WN18 used in [9] and WN11 used in [24]. Freebase is a large collaborative knowledge graph consisting of a large number of world facts, and we also employ two subsets from Freebase: FB15K used in [9] and FB13 used in [24]. The statistics of these datasets are listed in Table 2.

Table 2. Statistics of Datasets

Dataset	#Relation	#Entity	#Train	#Valid	#Test
WN18	18	40 943	141 442	5 000	5 000
FB15K	1 345	14 951	483 142	50 000	59 071
WN11	11	38 696	112 581	2 609	10 544
FB13	13	75 043	316 232	5 908	23 733

Note: "#" stands for "the number of".

5.2 Link Prediction

The goal of link prediction is to complete a triple (h, r, t) by predicting the missing entity h or t when given (r, t) or (h, r). Similar to [9, 22], we conduct the link prediction task on the same two datasets, i.e., WN18 and FB15K used in [9].

We adopt the identical evaluation methodology Hits@10 used in [9]. In the testing phase, for each test triple (h, r, t), we remove the head or tail entity and then replace it with the entity in the knowledge graph to construct a corrupted triple (h', r, t) or (h, r, t'). By ranking the entities in ascending order with respect to the dissimilarity scores calculated by the score function, we get the rank of the original triple in the sorted list. We report the proportion of correct entities ranked in the top 10, denoted as Hits@10. For the corrupted triples may also exist in the knowledge graphs, they should be regarded as correct triples. Therefore, we filter out the corrupted triples included in the train and valid datasets before ranking. This operation is denoted as "filter". Otherwise, it is denoted as "raw". It is clear that a good predictor should achieve a higher Hits@10.

Since the datasets we used are the same with our baselines (shown in Table 3) on this task, we directly compare their results reported in [9-11] with our results. During the training of our proposed method, we select learning rate λ in $\{0.1, 0.05, 0.001, 0.000\,5\}$, the margin γ in $\{1.0, 2.0, 4.0\}$, the embedding dimension of entities and relations d in $\{20, 50, 100, 200\}$, the rank k

Table 3. Evaluation Results on Link Prediction

Method	WN18 Hits@10(%)		FB15K Hits@10(%)	
	Raw	Filter	Raw	Filter
Unstructured	35.3	8.2	4.5	6.3
RESCAL	37.2	52.8	28.4	44.1
SE	68.5	80.5	28.8	39.8
SME(linear)	65.1	74.1	30.7	40.8
SME(bilinear)	54.7	61.3	31.3	41.3
LFM	71.4	81.6	26.0	33.1
TransE	75.4	89.2	34.9	47.1
TransH(unif)	75.4	86.7	42.5	58.5
TransH(bern)	73.0	82.3	45.7	64.4
TransR(unif)	78.3	91.7	43.8	65.5
TransR(bern)	79.8	92.0	48.2	68.7
PTransE(ADD, 2-step)	-	-	51.8	83.4
TransCoRe(unif)	81.0	94.6	51.1	80.5
TransCoRe(bern)	81.5	94.6	53.6	76.7

is an integer that belongs to the interval $(0, \min(d, N_r)]$, and the batch size B in $\{20, 120, 480, 1\,440, 4\,800\}$. The optimal parameters are determined on the validation set. 1) Under the "unif" setting, the optimal configuration is: $\lambda = 0.001$, $d = 100$, $k = 10$, $B = 1\,440$, $\gamma = 2.0$ and taking L_1 as dissimilarity on WN18; $\lambda = 0.000\,5$, $d = 200$, $k = 85$, $B = 1\,440$, $\gamma = 1.0$ and taking L_1 as dissimilarity on FB15K. 2) Under the "bern" setting, the optimal configuration is: $\lambda = 0.001$, $d = 100$, $k = 9$, $B = 1\,440$, $\gamma = 1.0$ and taking L_1 as dissimilarity on WN18; $\lambda = 0.000\,5$, $d = 200$, $k = 45$, $B = 1\,440$, $\gamma = 1.0$ and taking L_1 as dissimilarity on FB15K. For both datasets, we traverse all the training triples for 500 rounds.

Experimental results on both WN18 and FK15K are shown in Table 3. From the table we can conclude the followings. 1) On WN18, TransCoRe is effective and achieves the best performance compared with the baselines. Specifically, our method improves the accuracy of Hits@10 on "filter" by 5.4% compared with TransE, 2.9%(unif)/2.6%(bern), and achieves 94.6% on both "unif" and "bern" compared with TransR. 2) On FB15K, the performance of our proposed method is the best on "raw". Meanwhile, our method improves the accuracy of Hits@10 on "filter" by 15% on "unif" compared with TransR. Yet the method of PTransE[27] outperforms our method on "filter" for considering the relation paths that can provide a good supplement for knowledge graph embedding. Therefore, one piece of possible future work is to incorporate the relation paths in our proposed method just as PTransE.

For the comparison of Hits@10 of different kinds of relations, Table 4 shows the detailed results for relations with different mapping properties[1] on FB15K. From Table 4, we can see that TransCoRe is superior to the baselines.

5.3 Triple Classification

Triple classification is to predict whether a given triple (h, r, t) is positive or negative, which is a binary classification task on triples. We use three datasets in this task. The two datasets of WN11 and FB13 are released by [24]. Both datasets contain a small number of relations and we also choose the dataset of FB15K which contains more relations. Following the same methodology used in NTN, the evaluation of classification needs negative labels. The datasets of WN11 and FB13 already contain negative triples released by [24], and each corrupted triple is built from a golden triple. For FB15K, we follow the same way used in [24] to generate negative labels for WN11 and FB13. The performance of triple classification is evaluated as follows. For a triple (h, r, t), if the dissimilarity score obtained by f_r is less than a relation-specific threshold, then the triple is classified to be positive, and negative otherwise. The relation-specific threshold σ_r is determined by maximizing the classification accuracy on the validation dataset.

We directly use the experimental results of several baselines reported in [9-10, 24] since they use the same datasets of WN11 and FB13 on this task. As mentioned in [11], for a fair comparison, all reported results are without combination with word embedding. For the dataset of FB15K is built by ourselves, we use the code released by [24] for NTN and [11] for TransE, TransH and TransR to evaluate on FB15K.

Table 4. Evaluation Results on FB15K by Mapping Properties of Relations (%)

Method	Predicting Head (Hits@10)				Predicting Tail (Hits@10)			
	1-to-1	1-to-n	n-to-1	m-to-n	1-to-1	1-to-n	n-to-1	m-to-n
Unstructured	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
SE	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME(linear)	35.1	53.7	19.0	40.3	32.7	14.9	61.6	43.3
SME(bilinear)	30.9	69.6	19.9	38.6	28.2	13.1	76.0	41.8
TransE	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH(unif)	66.7	81.7	30.2	57.4	63.7	30.1	83.2	60.8
TransH(bern)	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransR(unif)	76.9	77.9	38.1	66.9	76.2	38.4	76.2	69.1
TransR(bern)	78.8	89.2	34.1	69.2	79.2	37.4	90.4	71.1
TransCoRe(unif)	84.5	82.9	65.7	84.7	85.4	56.0	83.9	82.4
TransCoRe(bern)	87.6	94.3	51.7	80.6	88.1	39.3	94.7	77.4

[1] Mapping properties of relations follows the same rules in [9].

For TransCoRe, we select learning rate λ in $\{0.01, 0.05, 0.001, 0.005\}$, the margin γ in $\{1.0, 2.0, 4.0\}$, the embedding dimension of entities and relations d in $\{20, 50, 100, 200\}$, and the rank k is an integer that belongs to the interval $(0, \min(d, N_r)]$, and the batch size B in $\{20, 120, 480, 1\,440, 4\,800\}$. The best configuration is determined according to the accuracy in validation dataset. 1) Under the "unif" setting, the optimal configuration is: $\lambda = 0.01$, $\gamma = 4.0$, $d = 20$, $k = 9$, $B = 480$ and taking L_1 as dissimilarity on WN11; $\lambda = 0.005$, $\gamma = 1.0$, $d = 100$, $k = 10$, $B = 480$ and taking L_1 as dissimilarity on FB13. 2) Under the "bern" setting, the optimal configuration is: $\lambda = 0.05$, $\gamma = 4.0$, $d = 20$, $k = 10$, $B = 480$ and taking L_1 as dissimilarity on WN11; $\lambda = 0.001$, $\gamma = 2.0$, $d = 100$, $k = 10$, $B = 480$ and taking L_1 as dissimilarity on FB13. For both datasets, we also traverse all the training triples for 500 rounds.

Experimental results of triple classification are shown in Table 5. On WN11, the results show that our method achieves a comparable performance with TransR and outperforms all the other baseline methods. This is because the category number of relations is small and the relationships among relations are weak on dataset WN11. On FB13, although the accuracy of TransCoRe is improved greatly than those of TransE, TransH and TransR, it still has almost two percentage points less than the most expressive model NTN. On the larger dataset of FB15K, 1 345 relations and 99.13% entities are directly linked by two or more distinct relations indicating the correlations among relations are much stronger. Results also show that TransE, TransH and TransR are much better than NTN, while TransCoRe performs the best.

Table 5. Evaluation Results of Triple Classification (%)

Method	WN11	FB13	FB15K
SE	53.0	75.2	-
SME(linear)	70.0	63.7	-
SLM	69.9	85.3	-
LFM	73.8	84.3	-
NTN	70.4	87.1	67.3
TransE(unif)	75.6	70.9	79.5
TransE(bern)	75.9	81.5	80.1
TransH(unif)	77.7	76.5	81.8
TransH(bern)	78.8	83.3	81.7
TransR(unif)	85.5	74.7	83.5
TransR(bern)	85.9	82.5	83.6
TransCoRe(unif)	86.1	76.0	87.9
TransCoRe(bern)	85.5	85.8	88.2

5.4 Discussions on the Correlations of Relations

In order to illustrate the effectiveness of correlations, we analyze the results for each relation on the task of link prediction on WN18 in detail. We plot the distribution of triples related to different relations as shown in Fig.4. It shows that the numbers of related triples vary sharply with relations. Specifically, the number of triples related to *_hyponym* is 34 832, while only 903 for *_synset_domain_region_of*. This phenomenon indicates that the triples for each relation are unevenly dis-

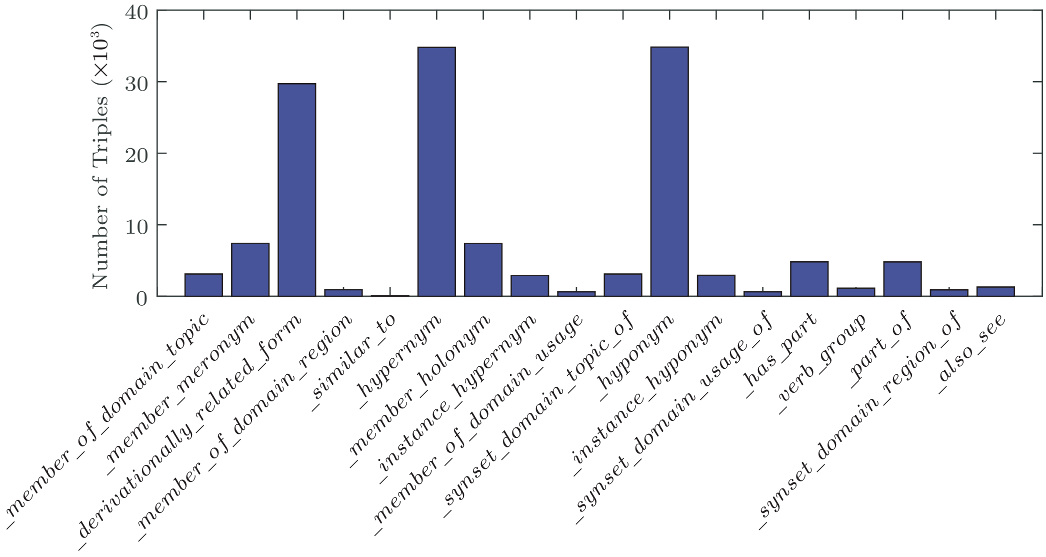


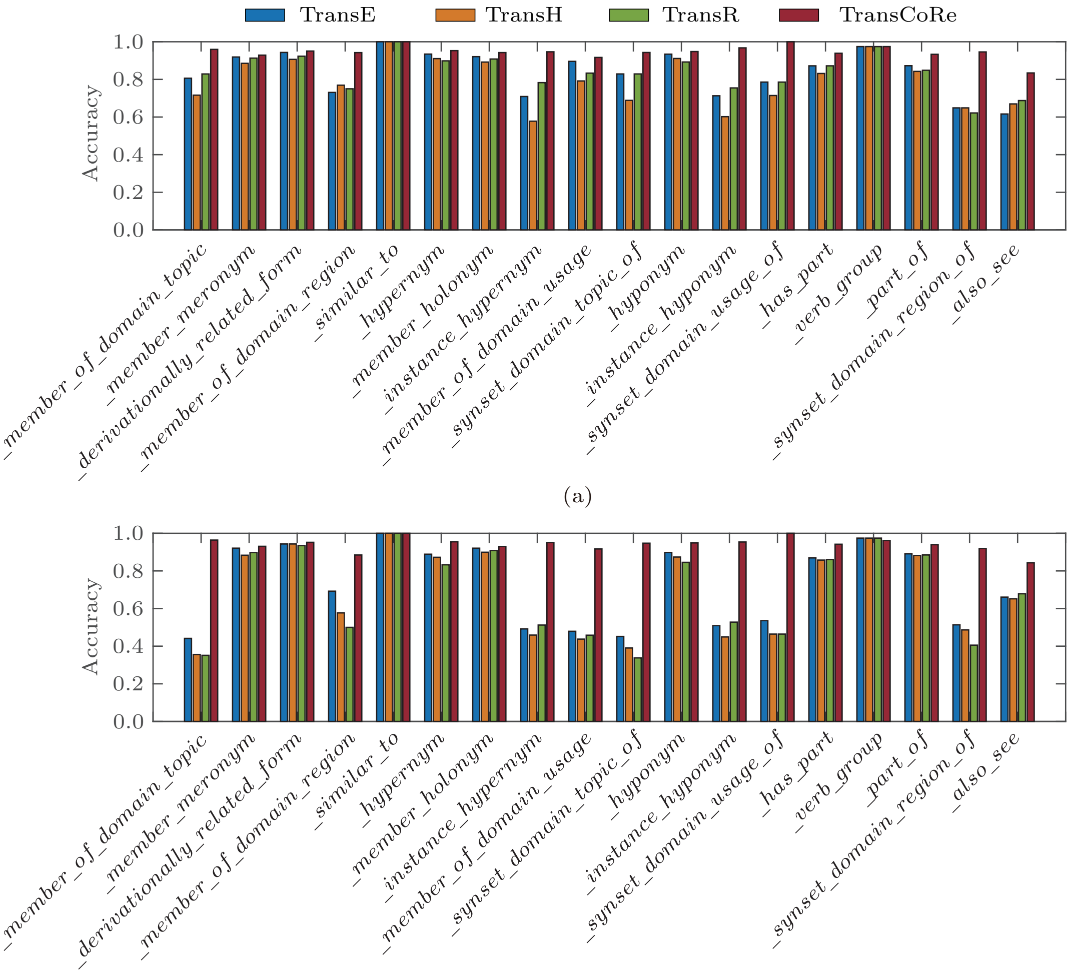
Fig.4. Triple distribution over relations.

tributed in the training data. Intuitively, the relation with sufficient triples will be learned better, and the performance on these relations may be better than that with limited triples on link prediction task.

We analyze the accuracies for each relation on the link prediction task by using TransE, TransH, TransR and TransCoRe, respectively. From the results shown in Fig.5, we can see that: 1) on the whole, the accuracies for each relation are increasing by the methods TransE, TransH, TransR and TransCoRe; 2) for each method, the accuracies for different relations vary sharply. In particular, under the "unif" case, the accuracies on TransE are fluctuated ranging from 61.6% to 100% (see Fig.5(a)). The similar phenomena can be observed when using the methods TransH and TransR. TransCoRe achieves the accuracies ranging from 83.4% to 100% in Fig.5(a) and the gap between the maximum accuracy and the minimum accuracy is narrower than that of the baselines. Under the "bern" case, similar findings can be observed in Fig.5(b).

An obvious reason for these phenomena is that the triples for each relation are unevenly distributed in the training data. For these baselines, the embeddings of different relations are learned separately. The embeddings for relations with adequate triples would be learned well, whereas for those with few triples would not. In our method, this issue can be addressed by leveraging the correlations of relations, for the learning process of each relation embedding is associated with the same matrix $\boldsymbol{U}$ which captures the common information of all relations. As a result, the relations with few triples could share the common information with other relations and better embeddings can be learned during training.

For further investigation, we continue to conduct experiments on part of training data which has few

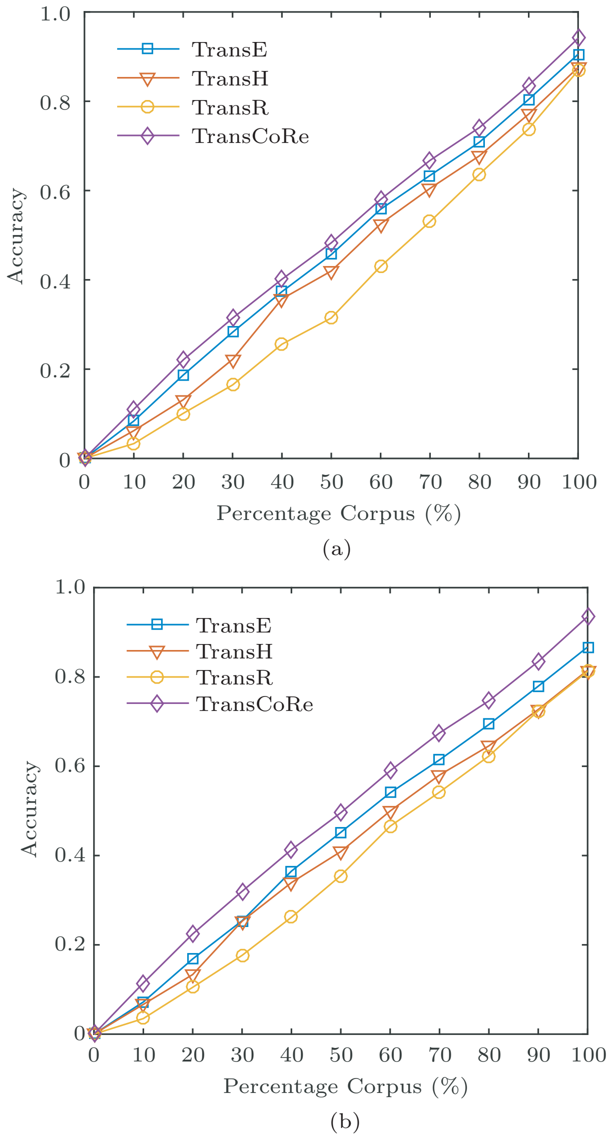


(a)

(b)

Fig.5. Comparisons on the results for each relation on link prediction under the settings of (a) "unif" and (b) "bern".

examples. For simplicity and without loss of generality, we choose the dataset WN18 and randomly extract 10%∼100% (stepped by 10%) triples from it as our experimental data. Afterwards, we use the code released by [11] to evaluate the methods of TransE, TransH and TransR on these datasets. For a fair comparison, all of the baseline methods and our method do not use pre-trained embeddings. The results are shown in Fig.6 with the configuration: $\lambda = 0.001$, $\gamma = 1$, $d = 80$, $B = 480$, and the rank k is set to 11 in our proposed method.



(a)

(b)

Fig.6. Results on link prediction by using 10%∼100% triples of WN18 under the settings of (a) "unif" and (b) "bern", respectively.

From Fig.6, we observe that our method consistently and significantly outperforms these baselines on all datasets under the settings of both "unif" and "bern". For instance, on the dataset containing 50% triples of WN18, our method surpasses TransE, TransH, and TransR on the prediction accuracy by 2.7%, 6.3% and 16.9% respectively. The main reason is that our method can capture the common information of all relations to learn better embeddings for the relations with few triples during training. On the contrary, these baselines learn embeddings for relations separately and would not get better embeddings for the relations without enough training data. In addition, an amazing find can be seen from the results is that TransR performs the worst, and the performance of TransE is better than that of both TransH and TransR. One reason is that no pre-trained embeddings are used to initialize the vectors of TransR. Moreover, the complexity of TransR is higher than that of TransE and TransH, which means that more corpora are needed for training.

6 Conclusions

In this paper, we firstly analyzed the embedded relation matrices learned by the typical baselines via applying SVD on them. Analysis results indicated that the correlations do exist among relations and they were shown as low-rank structure over the relation embedding vectors. Then, based on the analysis, we proposed a novel knowledge graph embedding method to learn better embeddings for entities and relations by capturing these correlations. Furthermore, extensive experiments conducted on the standard benchmark datasets of WordNet and Freebase demonstrated that our proposed method outperforms the typical baselines on the tasks of link prediction and triple classification. Finally, we investigated the effectiveness of correlations on the results of link prediction on WN18 in detail and the results proved that our proposed method can solve the unevenly distributed issues of triples and keep the improvements in the case of having few training data.

For future work, it might be interesting to apply the knowledge graph embedding technique to other important scenarios, e.g., text classification, document clustering, semantic question answering. For instance, it is worth leveraging knowledge graph for better representation of documents or texts.

References

[1] Miller G A. WordNet: A lexical database for English. *Communications of the ACM*, 1995, 38(11): 39-41.

[2] Bollacker K, Cook R, Tufts P. Freebase: A shared database of structured general human knowledge. In *Proc. the 22nd National Conf. Artificial Intelligence*, July 2007, pp.1962-1963.

[3] Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, June 2008, pp.1247-1250.

[4] Suchanek F M, Kasneci G, Weikum G. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proc. the 16th Int. World Wide Web Conf.*, May 2007, pp.697-706.

[5] Tang J, Lou T C, Kleinberg J, Wu S. Transfer learning to infer social ties across heterogeneous networks. *ACM Trans. Information Systems*, 2016, 34(2): Article No. 7.

[6] Jia Y T, Wang Y Z, Lin H L, Jin X L, Cheng X Q. Locally adaptive translation for knowledge graph embedding. In *Proc. the 30th AAAI Conf. Artificial Intelligence*, February 2016, pp.992-998.

[7] Wu W T, Li H S, Wang H X, Zhu K Q. Probase: A probabilistic taxonomy for text understanding. In *Proc. the ACM Int. Conf. Management of Data*, May 2012, pp.481-492.

[8] Jayaram N, Khan A, Li C K, Yan X F, Elmasri R. Querying knowledge graphs by example entity tuples. *IEEE Trans. Knowledge and Data Engineering*, 2015, 27(10): 2797-2811.

[9] Bordes A, Usunier N, Garcia-Durán A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In *Proc. the 26th Int. Conf. Neural Information Processing Systems*, December 2013, pp.2787-2795.

[10] Wang Z, Zhang J W, Feng J L, Chen Z. Knowledge graph embedding by translating on hyperplanes. In *Proc. the 28th AAAI Conf. Artificial Intelligence*, July 2014, pp.1112-1119.

[11] Lin Y K, Liu Z Y, Sun M S, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In *Proc. the 29th AAAI Conf. Artificial Intelligence*, January 2015.

[12] Alter O, Brown P O, Botstein D. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 2000, 97(18): 10101-10106.

[13] de Lathauwer L, de Moor B, Vandewalle J. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 2000, 21(4): 1253-1278.

[14] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In *Proc. the 26th Int. Conf. Neural Information Processing Systems*, December 2013, pp.3111-3119.

[15] Nickel M, Tresp V, Kriegel H P. Factorizing YAGO: Scalable machine learning for linked data. In *Proc. the 21st Int. Conf. World Wide Web*, April 2012, pp.271-280.

[16] Franz T, Schultz A, Sizov S, Staab S. TripleRank: Ranking semantic Web data by tensor decomposition. In *Proc. the 8th Int. Semantic Web Conf.*, October 2009, pp.213-228.

[17] Chang K W, Yih W T, Yang B S, Meek C. Typed tensor decomposition of knowledge bases for relation extraction. In *Proc. Conf. Empirical Methods in Natural Language Processing*, October 2014, pp.1568-1579.

[18] Chang K W, Yih W T, Meek C. Multi-relational latent semantic analysis. In *Proc. Conf. Empirical Methods in Natural Language Processing*, October 2013, pp.1602-1612.

[19] Kiers H A L. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 2000, 14(3): 105-122.

[20] Bordes A, Glorot X, Weston J, Bengio Y. Joint learning of words and meaning representations for open-text semantic parsing. In *Proc. the 15th Int. Conf. Artificial Intelligence and Statistics*, April 2012, pp.127-135.

[21] Bordes A, Glorot X, Weston J, Bengio Y. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 2014, 94(2): 233-259.

[22] Bordes A, Weston J, Collobert R, Bengio Y. Learning structured embeddings of knowledge bases. In *Proc. the 25th Int. Conf. Artificial Intelligence*, August 2011, pp.301-306.

[23] Jenatton R, Le Roux N, Bordes A, Obozinski G. A latent factor model for highly multi-relational data. In *Proc. the 25th Int. Conf. Neural Information Processing Systems*, December 2012, pp.3167-3175.

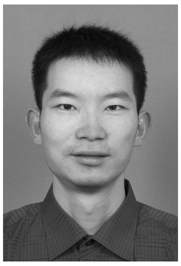
[24] Socher R, Chen D Q, Manning C D, Ng A. Reasoning with neural tensor networks for knowledge base completion. In *Proc. the 26th Int. Conf. Neural Information Processing Systems*, December 2013, pp.926-934.

[25] Pearson K. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 1895, 58(347/348/349/350/351/352): 240-242.

[26] Ji G L, He S Z, Xu L H, Liu K, Zhao J. Knowledge graph embedding via dynamic mapping matrix. In *Proc. the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, July 2015, pp.687-696.

[27] Lin Y K, Liu Z Y, Luan H B, Sun M S, Rao S W, Liu S. Modeling relation paths for representation learning of knowledge bases. In *Proc. Conf. Empirical Methods in Natural Language Processing*, September 2015, pp.705-714.

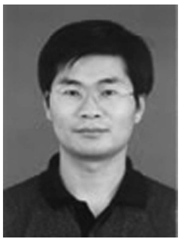
Ji-Zhao Zhu is now a Ph.D. candidate in the College of Computer Science and Engineering, Northeastern University, Shenyang. His research interests include knowledge graph, representation learning and parallel computation.



Yan-Tao Jia is an associate professor of Institute of Computing Technology, Chinese Academy of Sciences, Beijing. He received his Ph.D. degree in mathematics from Nankai University, Tianjin, in 2012. His main research interests include open knowledge network, social computing and combinatorial algorithms.

Jun Xu is a professor at the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. He received his B.E. and Ph.D. degrees in computer science from Nankai University, Tianjin, in 2001 and 2006, respectively. His major research interests focus on learning to rank for information retrieval. He has published more than 50 papers in international conferences and journals, including SIGIR, TOIS, and JMLR, etc. He currently serves as a senior program committee member for SIGIR and ACML.

Jian-Zhong Qiao received his B.E. degree in computer science from Xi'an Jiaotong University, Xi'an, in 1986, and his M.E. degree from Shenyang Institute of Computation Technology, Chinese Academy of Sciences, Shenyang, in 1991, and his Ph.D. degree in mechanical engineering and automation from Dalian University of Technology, Dalian, in 1999. He is currently a professor at the College of Computer Science and Engineering, Northeastern University, Shenyang. His research interests include parallel computation and artificial intelligence.



Xue-Qi Cheng is a professor at the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, and the director of the Key Laboratory of Network Data Science and Technology in ICT, CAS, Beijing. He received his Ph.D. degree in computer science from ICT, CAS, Beijing, in 2006. His main research interests include social computing, information security analysis, etc. He has published over 100 publications in prestigious journals and conferences, including TKDE, Physical Review E., SIGIR, WWW, etc. He has won the Best Paper Award in CIKM (2011) and the Best Student Paper Award in SIGIR (2012).