Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Knowledge Graph Completion with Adaptive Sparse Transfer Matrix

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{guoliang.ji, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn

Abstract

We model knowledge graphs for their completion by encoding each entity and relation into a numerical space. All previous work including Trans(E, H, R, and D) ignore the heterogeneity (some relations link many entity pairs and others do not) and the imbalance (the number of head entities and that of tail entities in a relation could be different) of knowledge graphs. In this paper, we propose a novel approach TranSparse to deal with the two issues. In TranSparse, transfer matrices are replaced by adaptive sparse matrices, whose sparse degrees are determined by the number of entities (or entity pairs) linked by relations. In experiments, we design structured and unstructured sparse patterns for transfer matrices and analyze their advantages and disadvantages. We evaluate our approach on triplet classification and link prediction tasks. Experimental results show that TranSparse outperforms Trans(E, H, R, and D) significantly, and achieves state-of-the-art performance.

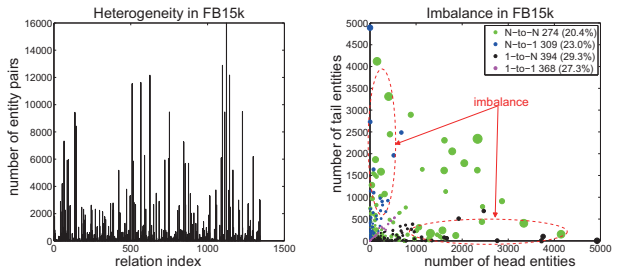
Introduction

Knowledge Graphs are directed graphs composed of entities as nodes and relations as edges. They store factual information in the form of triplet (*head, relation, tail*) (abbreviated as (h, r, t)), where *head* and *tail* are entities and *relation* represents the relationship from *head* to *tail*. Although the current knowledge graphs consist of large amounts of triplets, they are far from completeness, which is a pivotal parameter that determines the availability of a knowledge graph. This paper aims to provide an efficient, scalable approach to model knowledge graphs for their completion.

Recently, the volumes of existing knowledge graphs grow fast, which results in that the traditional logic-based approaches are intractable. To this end, embedding based approaches that encode each object in knowledge graphs into a continuous vector space show strong feasibility and robustness. Thus, this kind of approach has been attracting widespread attention. In general, existing embedding based approaches could be roughly divided into two groups, one is tensor factorization (Nickel, Tresp, and Kriegel 2011; 2012; Nickel and Tresp 2013a; 2013b; Nickel et al., 2015), the other is neural network models (Bordes et al., 2011; 2012; 2014; Socher et al., 2013; Wang et al., 2014; Lin et al., 2015; Ji et al., 2015; Guo et al., 2015). Tensor factorization approaches regard a knowledge graph as a three-way adjacency tensor, such as RESCAL. It factorizes the adjacency tensor into a core tensor and a factor matrix. Each entry in the factorization result would be regarded as a measure to indicate whether a corresponding triplet exists or not. Neural network approaches often define a score function with parameters and a margin-based training objective to punish negative triplets, such as Trans(E, H, R, and D).

Although these methods have strong ability to model knowledge graphs, it remains challenging for the reason that the objects (entities and relations) in a knowledge graph are **heterogeneous** and **unbalanced**. Specifically, (1) some relations link many entity pairs (called *complex relations*) and others do not (called *simple relations*); (2) some relations link many head (tail) entities and fewer tail (head) entities, such as the relation *gender* which can link many person names at head and only link *male*, *female* at tail. Figure 1 shows the statistics of the subgraph FB15k. From Figure 1, we can see that: (1) the complexities of different relations vary widely; and (2) unbalanced relations occupy a large proportion in a knowledge graph. All previous work including Trans(E, H, R, and D) do not consider the two issues, and they model each relation in the same way. Heterogeneity may lead to overfitting on simple relations or underfitting on complex relations. Meanwhile, imbalance of the two sides (head and tail) indicates that it is unreasonable to treat them (the two sides) equally.

Therefore, we argue that the relations with different complexities need different expressive models to learn, and the two sides should be modeled separately. In this paper, we use sparse matrices (another selection may be low-rank matrix, we will explain why we do not choose it later) to model the relations. To overcome the heterogeneity, we propose a model **TranSparse(share)**, in which the sparse degrees of transfer matrices are determined by the number of entity pairs linked by relations and the two sides of relations share the same transfer matrices. More specifically, the transfer matrices of complex relations would be less sparse than that of simple relations. Furthermore, to deal with the problem of imbalance of relations, we make a modification on TranSparse(share) and propose the second model **TranSparse(separate)**, in which each relation has two separate sparse transfer matrices, one for head and the other for



(a) Heterogeneity (b) Imbalance

Figure 1: Relation statistics of FB15k which contains 1,345 relations. (a) shows the heterogeneity of FB15k. In (b), each circle represents a relation. Area of the circle is proportional to the number of entity pairs linked by relations in train data.

tail. The sparse degrees are determined by the number of head (tail) entities linked by relations. In experiments, our approach is effective and outperforms all baseline models.

Our contributions are: (1) We propose a novel approach that considers the heterogeneity and the imbalance, which are not used in previous models, to embed knowledge graphs for their completion; (2) Our approach is efficient and has fewer parameters, which makes it easy to extend to large scale knowledge graphs; (3) We provide two sparse patterns for transfer matrices and analyze their advantages and disadvantages; and (4) In triplet classification and link prediction tasks, our approach achieves state-of-the-art performance.

Related Work

Before proceeding, we define our mathematical notations. We denote a triplet by (h, r, t) and their column vectors by bold lower case letters $\mathbf{h}, \mathbf{r}, \mathbf{t}$; matrices by bold upper case letters, such as $\mathbf{M}$. Score function is represented by $f_r(\mathbf{h}, \mathbf{t})$.

Translation-based Models

TransE (Bordes et al., 2013) regards the relation $\mathbf{r}$ as translation from $\mathbf{h}$ to $\mathbf{t}$ for a golden triplet (h, r, t). Hence, $(\mathbf{h} + \mathbf{r})$ is close to $(\mathbf{t})$ and the score function is $f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|^2_{\ell_{1/2}}$. It is effective, but only suitable for 1-to-1 relations, there remain flaws for 1-to-N, N-to-1 and N-to-N relations.

TransH (Wang et al., 2014) regards relation as a translating operation on a relation-specific hyperplane, which is characterized by a norm vector $\mathbf{w}_r$ and a translation vector $\mathbf{d}_r$. The embeddings $\mathbf{h}$ and $\mathbf{t}$ are projected to the hyperplane of relation r to obtain projected vectors $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. The socre function is $f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|^2_{\ell_{1/2}}$.

TransR/CTransR (Lin et al., 2015) models entity and relation in different vector spaces. It sets a transfer matrix $\mathbf{M}_r$ for each relation r to map entity embedding to relation vector space. Its score function is $f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{M}_r\mathbf{h} + \mathbf{r} - \mathbf{M}_r\mathbf{t}\|^2_{\ell_{1/2}}$. In TransR/CTransR, simple relations may be **overfitting** or complex relation may be **underfitting** because every relation (no matter complex or simple) has the same number of parameters to learn.

TransD (Ji et al., 2015) considers the multiple types of entities and relations simultaneously, and replaces transfer matrix by the product of two projection vectors of an entity-relation pair. TransD obtains state-of-the-art performance on triplet classification and link prediction tasks.

KG2E (He et al., 2015) uses Gaussian embedding to model the data uncertainty. It has the best performs on 1-to-N and N-to-1 relations at present.

Other Methods

Structured Embedding (SE). SE (Bordes et al., 2011) sets two separate matrices $\mathbf{M}_{rh}$ and $\mathbf{M}_{rt}$ for each relation to project head and tail entities, respectively. Its score function is $f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{M}_{rh}\mathbf{h} - \mathbf{M}_{rt}\mathbf{t}\|_1$.
Semantic Matching Energy (SME). (Bordes et al., 2012; 2014) encodes each entity and relation into a vector. Its score function is a neural network that captures correlations between entities and relations via linear algebra operations. Parameters of the neural network are shared by all relations.
Latent Factor Model (LFM). LFM (Jenatton et al., 2012; Sutskever, Tenenbaum, and Salakhutdinov, 2009) encodes each entity as a vector and sets a matrix $\mathbf{M}_r$ for each relation. It defines the score function as $f_r(\mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t}$.
Single Layer Model (SLM). The model constructs a nonlinear neural network to represent the score function. It is designed as a baseline of NTN model (Socher et al., 2013).
Neural Tensor Network (NTN). NTN (Socher et al., 2013) extends SLM model by considering the second-order correlations into nonlinear neural networks.

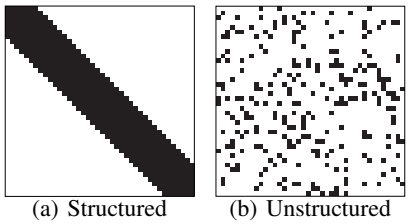
In experiments, we compare our approach with the above models. We also report the **RESCAL** system's performance presented in Wang et al. (2014) and Lin et al. (2015).

Our Model

All the approaches presented in Related Work do not consider the heterogeneity and the imbalance, which are crucial difficulties of knowledge graphs embedding. Here, we propose a new approach TranSparse that uses adaptive sparse matrices to model different types of relations.

Sparse Matrix

Sparse matrices refer to the matrices in which most entries are zeros. The fraction of zero elements over the total number of elements in a matrix is called **sparse degree** (denoted by θ). We use $\mathbf{M}(\theta)$ to denote a matrix $\mathbf{M}$ with sparse degree θ. Sparse matrix is by nature more easily compressed and thus requires less storage. Furthermore, only the nonzero elements involve calculation, which can reduce the computational cost drastically. Figure 2 shows two typical sparse matrices: structured and unstructured (Saad 2003). In Figure 2, nonzero entries in structured pattern locate on the diagonal band, while for unstructured pattern, they locate with an uniform random distribution. There is an important distinction between the two patterns: the structured pattern is conductive to matrix-by-vector products, but the unstructured pattern is not (Saad 2003). Therefore, structured pattern can make our model scale up to large knowledge graphs more easily. However, unstructured pattern is more generally used



(a) Structured (b) Unstructured

Figure 2: Sparse Patterns: (a) Structured. (b) Unstructured. Black squares represent the nonzero elements. The two sparse matrices have the same number of nonzero entries.

in other literatures, i.e., Candès et al. (2011). And it often brings better experimental results than structured pattern[1]. We will use the two sparse patterns in our experiments and compare their performances in several datasets.

Sparse Matrix vs Low-Rank Matrix

Based on our motivations, we need to use matrices with higher and lower degrees of freedom to learn complex and simple relations, respectively. The degrees of freedom of a weight matrix refers to the number of variates which are independence. For a weight matrix $\mathbf{M}$, the low-rank and sparsity both can reduce the degrees of freedom beacuse they are both constrains enforced on $\mathbf{M}$. Specifically, low-rank enforces some variables to satisfy specific constraints so that the variables in $\mathbf{M}$ can't be assigned freely. Thus, the degrees of freedom is reduced. For sparse matrices, we let some elements are zeros and don't change their values during training, and the other entries are freedom variates which can be learned by training. Thus, the degrees of freedom is the number of variates learned by training. But the sparse matrix is more suitable for our tasks for the two reasons.

- Sparse matrix is more flexible than low-rank matrix. We assume $\mathbf{M} \in \mathbb{R}^{m\times n}$, then $rank(\mathbf{M}) \leq \min(m, n)$. If we use low-rank to control the degrees of freedom, we can only obtain $\min(m, n)$ low-rank matrices for relations because the rank can determine the degrees of freedom of a matrix[2]. However, if we use the sparsity to control the degrees of freedom, we can obtain $m \times n$ sparse matrices because $\mathbf{M}$ contains $m \times n$ elements. Thus, for the dataset which has many relations, such as FB15k (contains 1,345 relations), sparsity is more flexible than low-rank.
- Sparse matrix is more efficient than low-rank matrix. In sparse matrix, only the nonzero entries involve calculation, which reduces the computational cost significantly. But, the low-rank matrix doesn't have the advantage. Therefore, with sparse matrix, our model can extend to large-scale knowledge graphs more easily.

[1]The structured sparse pattern only provides fixed and local linear combinations. However, the unstructured sparse pattern provides flexible and long-range linear combinations. Therefore, the later may learn better representations for KGs.

[2]A $m\times n$ rank-k matrix has $k(m+n)-k^2$ degrees of freedom. The proof can be found in Kennedy and Examination II (2013).

TranSparse

In previous work, no matter simple or complex, each relation has a transfer matrix which has the same number of parameters. As described in Introduction, complex relations link more head or tail entities, and they are more meaningful than simple relations. Therefore, complex relations need more parameters to learn fully and simple relations need fewer parameters to learn properly, which can stop the model from underfitting or overfitting. As the transfer matrices in our model are all sparse matrices, we call it TranSparse. We argue that the complexity of a relation is proportional to the number of triplets (or entities) linked by it because the more data linked by the relation, the more knowledge it contains.

TranSparse(share) In TranSparse(share), we set a sparse transfer matrix $\mathbf{M}_r(\theta_r)$ and a translation vector $\mathbf{r}$ for each relation r. N_r represents the number of entity pairs linked by relation r and N_{r^*} denotes the maximum number of them (relation r^* links the most entity pairs). We set a minimum sparse degree $\theta_{min}(0 \leq \theta_{min} \leq 1)$[3] for the matrix $\mathbf{M}_{r^*}$. Then the sparse degrees of transfer matrices are defined as

$$\theta_r = 1 - (1 - \theta_{min})N_r/N_{r^*}. \tag{1}$$

With the sparse transfer matrices, the projected vectors can be obtained by

$$\mathbf{h}_p = \mathbf{M}_r(\theta_r)\mathbf{h}, \quad \mathbf{t}_p = \mathbf{M}_r(\theta_r)\mathbf{t}. \tag{2}$$

where $\mathbf{M}_r(\theta_r) \in \mathbb{R}^{m\times n}$, $\mathbf{h}, \mathbf{t} \in \mathbb{R}^n$ and $\mathbf{h}_p, \mathbf{t}_p \in \mathbb{R}^m$.

TranSparse(separate) Here, we set two separate sparse matrices $\mathbf{M}_r^h(\theta_r^h)$ and $\mathbf{M}_r^t(\theta_r^t)$ for each relation, where the subscript r is the index of relations; h, t means which entity (head or tail) the matrix is used for. N_r^l $(l = h, t)$ denotes the number of entities linked by relation r at location l and $N_{r^*}^{l^*}$ denotes the maximum number of them (relation r^* links the most entities at location l^*). We set a minimum sparse degree $\theta_{min}(0 \leq \theta_{min} \leq 1)$ for the matrix $\mathbf{M}_{r^*}^{l^*}$. Then the sparse degrees of transfer matrices are

$$\theta_r^l = 1 - (1 - \theta_{min})N_r^l/N_{r^*}^{l^*} \quad (l = h, t). \tag{3}$$

The projected vectors of entities are defined as follows:

$$\mathbf{h}_p = \mathbf{M}_r^h(\theta_r^h)\mathbf{h}, \quad \mathbf{t}_p = \mathbf{M}_r^t(\theta_r^t)\mathbf{t}. \tag{4}$$

where $\mathbf{M}_r^h(\theta_r^h), \mathbf{M}_r^t(\theta_r^t) \in \mathbb{R}^{m\times n}$.

Both in TranSparse(share, separate), the score function is

$$f_r(\mathbf{h}, \mathbf{t}) = \|\mathbf{h}_p + \mathbf{r} - \mathbf{t}_p\|_{\ell_{1/2}}^2. \tag{5}$$

where $\mathbf{r} \in \mathbb{R}^m$. The score is expected to be lower for a golden triplet and higher for an incorrect triplet.

Training Objective

We denote the i-th triplet by $(h_i, r_i, t_i)(i = 1, 2, \ldots)$. Each triplet has a label y_i to indicate the triplet is positive $(y_i = 1)$ or negative $(y_i = 0)$. Then the positive and negative sets are denoted by $\Delta = \{(h_i, r_i, t_i) \mid y_i = 1\}$ and $\Delta' = \{(h_i, r_i, t_i) \mid y_i = 0\}$, respectively. An important trouble is that knowledge graphs only encode positive training triplets,

[3]θ_{min} is a hyper-parameter.

they do not contain negative examples. Therefore, we construct Δ' as follows: $\Delta' = \{(h'_i, r_i, t_i) \mid h'_i \neq h_i \wedge y_i = 1\} \cup \{(h_i, r_i, t'_i) \mid t'_i \neq t_i \wedge y_i = 1\}$. We also use the two strategies "unif" and "bern" described in (Wang et al., 2014) to replace the head or tail entity.

We use (h, r, t) to represent a positive triplet. We select one negative triplet for each positive triplet and denote it by (h', r, t') (replace $head$ or $tail$). Then we define the following margin-based ranking loss as the objective for training:

$$L = \sum_{(h,r,t)\in\Delta} \sum_{(h',r,t')\in\Delta'} [\gamma + f_r(\mathbf{h}, \mathbf{t}) - f_r(\mathbf{h}', \mathbf{t}')]_+ \quad (6)$$

where $[x]_+ \triangleq \max(0, x)$, and γ is the margin separating positive triplets and negative triplets. In practice, we enforce the following constrains: $\|\mathbf{h}\|_2 \leq 1, \|\mathbf{r}\|_2 \leq 1, \|\mathbf{t}\|_2 \leq 1, \|\mathbf{h}_p\|_2 \leq 1, \|\mathbf{t}_p\|_2 \leq 1$.

Algorithm Implementation

In order to speed up the convergence and avoid overfitting, we initiate entity and relation embeddings with the results of TransE. Thus, we should let the transfer matrices be square (the number of rows and that of columns are the same) and initiate them with identity matrices. Please note that, these are not necessary (one can use random initialization and let transfer matrices not be square).

For a transfer matrix $\mathbf{M}(\theta) \in \mathbb{R}^{n\times n}$, it has $nz = \lfloor \theta \times n \times n \rfloor$($\lfloor x \rfloor$ returns the maximum integer no more than x) nonzero elements. As we initialize the transfer matrix with identity matrix (the diagonal are nonzero), the number of other nonzero entries is $nz' = nz - n$, and when $nz \leq n$, we set $nz' = 0$ (the transfer matrix is an identity matrix).

When we construct structured pattern for the transfer matrix $\mathbf{M}(\theta)$, we let the nz' nonzero entries locate on the two sides of diagonal and be symmetry along the diagonal (see structured pattern shown in Figure 2). If the number nz' can't satisfy this requirements, we select another integer, which is near to nz', to replace it. In some cases, this strategy may change the number of nonzero entries a little, but it doesn't affect our experiments significantly. When we construct unstructured pattern, we only scatter the nz' nonzero elements in $\mathbf{M}(\theta)$ randomly (but not on the diagonal).

Before training, we first set the hyper-parameter θ_{min} and compute sparse degrees for all transfer matrices. Then we construct sparse transfer matrices with structured and unstructured patterns for our models. We only update the nonzero entries during training. The details of optimization procedure of TranSparse is described in Algorithm 1.

Comparisons of complexity

Table 1 lists the complexities of several neural network models described in Related Work. The average sparse degree $\hat{\theta}$ is near to 1 in practice. Therefore, the complexity (both on the number of parameters and the times of multiplication operations) of TranSparse is similar to TransH and TransD, which shows the high efficiency of our approach.

Algorithm 1 Learning TranSparse(separate).

Require:
Training sets Δ and Δ', entity set E and relation set R, margin γ, embeddings dim n, θ_{min} and learning rate α. Two set lists $LIST_{head} = [S_{h,1}, \cdots, S_{h,r}, \cdots, S_{h,|R|}]$ and $LIST_{tail} = [S_{t,1}, \cdots, S_{t,r}, \cdots, S_{t,|R|}]$, where $S_{a,b}$ ($a = h, t$ and b is the index of relations) is a set of index pairs $(row, column)$ which indicate the locations of nonzero entries in transfer matrix $\mathbf{M}_b^a$. The two set lists are used to find nonzero entries in transfer matrix during training.

Ensure:
Entity and relation embeddings. All sparse transfer matrices.

initialize Running TransE, and initializing each entity $e \in E$ and relation $r \in R$ with the results of TransE. Let each transfer matrix be an identity matrix.
loop
  $\Delta_{batch} \leftarrow$ sample (Δ, b) //sample a minibatch of size b
  $T_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
  for $(h, r, t) \in \Delta_{batch}$ **do**
    $(h', r, t') \leftarrow$ sample $(\Delta'_{(h,r,t)})$ //sample a corrupted triplet
    $T_{batch} \leftarrow T_{batch} \cup \{(h, r, t), (h', r, t')\}$
  end for
  Update embeddings and nonzero entries in transfer matrices w.r.t.
  $$\sum_{((h,r,t),(h',r,t'))\in T_{batch}} \nabla[\gamma + f_r(\mathbf{h}, \mathbf{t}) - f_r(\mathbf{h}', \mathbf{t}')]_+$$
  for $\ell \in$ entities or relations in T_{batch} **do**
    if $\|\boldsymbol{\ell}\|_2 > 1$ **then**
      $\boldsymbol{\ell} \leftarrow \boldsymbol{\ell}/\|\boldsymbol{\ell}\|_2$ //constrains:$\|\mathbf{h}\|_2 \leq 1, \|\mathbf{r}\|_2 \leq 1, \|\mathbf{t}\|_2 \leq 1$
    end if
  end for
  Each triplet has three projected vectors ($\mathbf{h}_p, \mathbf{t}_p, \mathbf{h}'_p$ or $\mathbf{t}'_p$). We denote the projected vectors of the i-th triplet in T_{batch} by $\boldsymbol{\ell}^i_{p,1}, \boldsymbol{\ell}^i_{p,2}$ and $\boldsymbol{\ell}^i_{p,3}$. Then we define $\mathcal{L} = \sum_{i=1}^{b} \sum_{j=1}^{3} \max\{\|\boldsymbol{\ell}^i_{p,j}\|_2 - 1, 0\}$.
  while $\mathcal{L} > 0$ **do**
    Update embeddings and nonzero entries in transfer matrices w.r.t. $\nabla\mathcal{L}$. //coinstrains: $\|\mathbf{h}_p\|_2 \leq 1, \|\mathbf{t}_p\|_2 \leq 1$.
  end while
end loop

Experiments and Analysis

In this section, we choose triplet classification and link prediction tasks to evaluate our approach.

Datasets

We do triplet classification and link prediction tasks on WordNet (Miller 1995) and Freebase (Bollacker et al., 2008). WordNet is a large lexical knowledge graph. Entities in WordNet are synonyms which express distinct concepts. Relations in WordNet are conceptual-semantic and lexical relations. In this paper, we use two subsets of WordNet: WN11 (Socher et al., 2013) and WN18 (Bordes et al., 2014). Freebase is a large collaborative knowledge base which consists of large amounts of facts. We also use two subsets of Freebase: FB15k (Bordes et al., 2014) and FB13 (Socher et al., 2013). Table 2 lists the statistics of the four datasets.

Triplet Classification

Triplet classification aims to judge whether a given triplet (h, r, t) is correct or not, which is a binary classification task. In this paper ,we use three datasets WN11, FB13 and FB15k to evaluate our models. The test sets of WN11 and FB13 provided by (Socher et al., 2013) contain positive and negative triplets. As to FB15k, its test set only contains correct

Model	#Parameters	# Operations (Time complexity)
SLM (Socher et al. 2013)	$O(N_e m + N_r(2k + 2nk))(m = n)$	$O((2mk + k)N_t)$
NTN (Socher et al. 2013)	$O(N_e m + N_r(n^2 s + 2ns + 2s))(m = n)$	$O(((m^2 + m)s + 2mk + k)N_t)$
TransE (Bordes et al. 2013)	$O(N_e m + N_r n)(m = n)$	$O(N_t)$
TransH (Wang et al. 2014)	$O(N_e m + 2N_r n)(m = n)$	$O(2mN_t)$
TransR (Lin et al. 2015)	$O(N_e m + N_r(m + 1)n)$	$O(2mnN_t)$
CTransR (Lin et al. 2015)	$O(N_e m + N_r(m + d)n)$	$O(2mnN_t)$
TransD (Ji et al. 2015)	$O(2N_e m + 2N_r n)$	$O(2nN_t)$
TranSparse(share) (this paper)	$O(N_e m + N_r(1 - \hat{\theta})(m + 1)n)(0 \ll \hat{\theta} \leq 1)$	$O(2(1 - \hat{\theta})mnN_t)(0 \ll \hat{\theta} \leq 1)$
TranSparse(separate) (this paper)	$O(N_e m + 2N_r(1 - \hat{\theta})(m + 1)n)(0 \ll \hat{\theta} \leq 1)$	$O(2(1 - \hat{\theta})mnN_t)(0 \ll \hat{\theta} \leq 1)$

Table 1: Complexities (the number of parameters and the times of multiplication operations in an epoch) of several embedding models. N_e and N_r represent the number of entities and relations, respectively. N_t represents the number of triplets in a knowledge graph. m is the dimension of entity embedding space and n is the dimension of relation embedding space. d denotes the number of clusters of a relation. k is the number of hidden nodes of a neural network and s is the number of slice of a tensor. $\hat{\theta}(0 \ll \hat{\theta} \leq 1)$ denotes the average sparse degree of all transfer matrices.

Dataset	#Rel	#Ent	#Train	#Valid	#Test
WN11	11	38,696	112,581	2,609	10,544
WN18	18	40,493	141,442	5,000	5,000
FB13	13	75,043	316,232	5908	23,733
FB15k	1,345	14,951	483,142	50,000	59,071

Table 2: Datasets used in our experiments.

triplets. We construct negative triplets for FB15k following the same setting used for FB13 (Socher et al., 2013).

For triplet classification, we set a threshold δ_r for each relation r. δ_r is obtained by maximizing the classification accuracies on the valid set. For a given triplet (h, r, t), if its score is lower than δ_r, it will be classified as positive, otherwise negative.

In this experiments, we select the margin γ among $\{1, 1.5, 2, 4, 10\}$, the learning rate λ for SGD (Duchi, Hazan, and Singer 2011) among $\{0.1, 0.01, 0.001\}$, the minimum sparse degree θ_{min} among $\{0.0, 0.1, 0.3, 0.7, 0.9\}$, the dimension of vectors n amon g $\{20, 50, 80, 100\}$, and the mini-batch size B among $\{100, 200, 1000, 4800\}$. The best configuration obtained by valid set are: $\gamma = 4, \lambda = 0.001, \theta_{min} = 0.7, n = 20, B = 1000$ and taking L_1 as dissimilarity on WN11; $\gamma = 1, \lambda = 0.001, \theta_{min} = 0.9, n = 100, B = 200$ and taking L_2 as dissimilarity on FB13; $\gamma = 1.5, \lambda = 0.001, \theta_{min} = 0.0, n = 100, B = 4800$ and taking L_1 as dissimilarity on FB15k.

Comparisions We compare our approach with the models described in Related Work. As we construct negative triplets for FB15k by ourselves, we revaluate the dataset instead of reporting the results of (Wang et al., 2014; Lin et al., 2015) directly. Table 3 shows the evaluation results of triplet classification. On WN11, TranSparse obtains the accuracy of 86.8%, which outperforms all the baseline models. On FB13, all the accuracies of our method are higher than that of Trans(E, H, and R) significantly, and they are near to the best accuracy of 89.1% of TransD. On FB15k, our approach also achieves state-of-the-art result by 88.5%.

[4]We report the average results of 5 times for "US" to ensure the results are more believable.

Data sets	WN11	FB13	FB15k
SE	53.0	75.2	-
SME(bilinear)	70.0	63.7	-
SLM	69.9	85.3	-
LFM	73.8	84.3	-
NTN	70.4	87.1	68.2
TransE(unif / bern)	75.9 / 75.9	70.9 / 81.5	77.3 / 79.8
TransH(unif / bern)	77.7 / 78.8	76.5 / 83.3	74.2 / 79.9
TransR(unif / bern)	85.5 / 85.9	74.7 / 82.5	81.1 / 82.1
CTransR(bern)	85.7	-	84.3
TransD(unif / bern)	85.6 / 86.4	85.9 / **89.1**	86.4 / 88.0
TranSparse(share, S, unif / bern)	86.2 / 86.3	85.5 / 87.8	85.7 / 87.9
TranSparse(share, US, unif / bern)	86.3 / 86.3	85.3 / 87.7	86.2 / 88.1
TranSparse(separate, S, unif / bern)	86.2 / 86.4	86.7 / **88.2**	87.1 / **88.3**
TranSparse(separate, US, unif / bern)	**86.8 / 86.8**	86.5 / 87.5	87.4 / **88.5**

Table 3: Experimental results of Triplet Classification(%). "S" and "US" represent the structured and unstructured sparse patterns, respectively[4].

Analysis (1) In Table 3, TranSparse(share) obtains better performance than Trans(E, H, R) and is near to TransD, which indicates that the heterogeneity is important in our tasks. In Figure 3, TranSparse improves the performance of TransR both on simple and complex relations. For the reason, we assume that TranSparse has fewer parameters for simple relations and more parameters for complex relations than TransR. Therefore, TranSparse deals with the data heterogeneity better; (2) TranSparse(separate) obtains better performance than TranSparse(share), which illustrates that the imbalance is also a crucial issue and our method can reduce its negative impacts; (3) Unstructured pattern often works slightly better than structured patter. It may be that unstructured pattern is closer to the optimum sparse pattern.

Link Prediction

Link prediction is to predict the missing h or t for a golden triplet (h, r, t). In this task, we remove the head or tail entity and then replace it with all the entities in dictionary in turn for each triplet in test set. We first compute scores of

Data sets	WN18				FB15K			
Metric	Mean Rank		Hits@10		Mean Rank		Hits@10	
	Raw	Filt	Raw	Filt	Raw	Filt	Raw	Filt
RESCAL (Nickle, Tresp, and Kriegel 2011)	1,180	1,163	37.2	52.8	828	683	28.4	44.1
SE (Bordes et al. 2011)	1,011	985	68.5	80.5	273	162	28.8	39.8
SME (linear / bilinear) (Bordes et al.2012)	545 / 526	533 / 509	65.1 / 54.7	74.1 / 61.3	274 / 284	154 / 158	30.7 / 31.3	40.8 / 41.3
LFM (Jenatton et al. 2012)	469	456	71.4	81.6	283	164	26.0	33.1
TransE (Bordes et al. 2013)	263	251	75.4	89.2	243	125	34.9	47.1
TransH (unif / bern) (Wang et al. 2014)	318 / 401	303 / 388	75.4 / 73.0	86.7 / 82.3	211 / 212	84 / 87	42.5 / 45.7	58.5 / 64.4
TransR (unif / bern) (Lin et al. 2015)	232 / 238	219 / 225	78.3 / 79.8	91.7 / 92.0	226 / 198	78 / 77	43.8 / 48.2	65.5 / 68.7
CTransR (unif / bern) (Lin et al. 2015)	243 / 231	230 / 218	78.9 / 79.4	92.3 / 92.3	233 / 199	82 / 75	44.0 / 48.4	66.3 / 70.2
TransD (unif / bern) (Ji et al. 2015)	242 / **224**	229 / **212**	79.2 / 79.6	92.5 / 92.2	211 / 194	67 / 91	49.4 / 53.4	74.2 / 77.3
TranSparse (share, S, unif / bern)	248 / 237	236 / 224	79.7 / **80.4**	93.5 / 93.6	226 / 194	95 / 88	48.8 / 53.4	73.4 / 77.7
TranSparse (share, US, unif / bern)	242 / 233	229 / 221	79.8 / **80.5**	**93.7 / 93.9**	231 / 191	101 / 86	48.9 / **53.5**	73.5 / 78.3
TranSparse (separate, S, unif / bern)	235 / **224**	223 / 221	79.0 / 79.8	92.3 / 92.8	211 / **187**	**63** / 82	50.1 / 53.3	77.9 / **79.5**
TranSparse (separate, US, unif / bern)	233 / **223**	221 / **211**	79.6 / 80.1	93.4 / 93.2	216 / **190**	**66** / 82	50.3 / **53.7**	78.4 / **79.9**

Table 4: Experimental results of link prediction. "S" and "US" represent structured and unstructured pattern, respectively (%).

Tasks	Prediction Head (Hits@10)				Prediction Tail (Hits@10)			
Relation Category	1-to-1	1-to-N	N-to-1	N-to-N	1-to-1	1-to-N	N-to-1	N-to-N
SE (Bordes et al. 2011)	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME (linear / Bilinear) (Bordes et al.2012)	35.1 / 30.9	53.7 / 69.6	19.0 / 19.9	40.3	32.7 / 38.6 / 28.2	14.9 / 13.1	61.6 / 76.0	43.3 / 41.8
TransE (Bordes et al. 2013)	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH (unif / bern) (Wang et al. 2014)	66.7 / 66.8	81.7 / 87.6	30.2 / 28.7	57.4 / 64.5	63.7 / 65.5	30.1 / 39.8	83.2 / 83.3	60.8 / 67.2
TransR (unif / bern) (Lin et al. 2015)	76.9 / 78.8	77.9 / 89.2	38.1 / 34.1	66.9 / 69.2	76.2 / 79.2	38.4 / 37.4	76.2 / 90.4	69.1 / 72.1
CTransR (unif / bern) (Lin et al. 2015)	78.6 / 81.5	77.8 / 89.0	36.4 / 34.7	68.0 / 71.2	77.4 / 80.8	37.8 / 38.6	78.0 / 90.1	70.3 / 73.8
TransD (unif / bern) (Ji et al. 2015)	80.7 / 86.1	85.8 / 95.5	47.1/ 39.8	75.6 / 78.5	80.0 / 85.4	54.5/ 50.6	80.7 / **94.4**	77.9 / 81.2
TranSparse(share, S, unif / bern)	83.2 / **87.5**	86.4 / **95.9**	50.3 / 44.1	73.9 / 78.7	84.8 / **87.6**	57.7 / 55.6	83.3 / 93.9	75.3 / 80.6
TranSparse(share, US, unif / bern)	83.4 / **87.1**	86.7 / **95.8**	49.8 / 44.2	73.4 / 79.1	84.8 / 87.2	57.3 / 55.5	78.2 / 94.1	76.4 / 81.7
TranSparse(separate, S, unif / bern)	82.3 / 86.8	85.2 / 95.5	**51.3** / 44.3	79.6 / **80.9**	82.3 / 86.6	**59.8** / 56.6	84.9 / **94.4**	82.1 / **83.3**
TranSparse(separate, US, unif / bern)	83.2 / **87.1**	85.2 / **95.8**	**51.8** / 44.4	80.3 / **81.2**	82.6 / **87.5**	**60.0** / 57.0	85.5 / **94.5**	82.5 / **83.7**

Table 5: Experimental results of FB15k by mapping properties of relations (%).

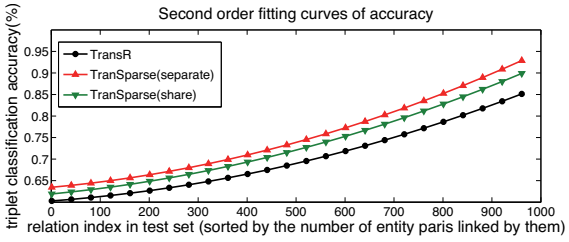


Figure 3: Second order fitting curves of accuracy of several embedding models on FB15k. There are 961 relations in test set. The horizontal axis represents relation index sorted by the number of entity pairs they link in train set, and the vertical axis represents the triplet classification accuracy.

those corrupted triplets and then rank them by ascending order; then the rank of the correct entity is finally stored. The task emphasizes the rank of the correct entity instead of only finding the best one entity. We report two measures as our evaluation metrics: the average rank of all correct entities (*Mean Rank*) and the proportion of correct entities ranked in top 10 (*Hits@10*). We call the evaluation setting "Raw". Noting the fact that a corrupted triplet may also exist in the knowledge graph, the corrupted triplet should be regarded as a correct triplet (Bordes et al., 2013). Hence, we remove the corrupted triplets included in train, valid and test sets before ranking. We call this evaluation setting "Filter". In this paper, we will report evaluation results of the two settings.

In this task, we use two datasets: WN18 and FB15k. We select the margin γ among $\{1, 2, 3.5, 4\}$, the learning rate λ for SGD among $\{0.1, 0.01, 0.001\}$, the minimum sparse degree θ_{min} among $\{0.0, 0.3, 0.5, 0.8\}$, the dimension of vectors n among {20, 50, 80, 100}, and the mini-batch size B among {100, 200, 1400, 4800}. On WN18, the best configuration obtained by valid set are: $\gamma = 3.5, \theta_{min} = 0.0, \lambda = 0.01, n = 50, B = 1400$ and taking L_1 as dissimilarity.

Comparisions Experimental results on both WN18 and FB15k are shown in Tabel 4. On WN18, our approach improves the accuracy of *Hits@10* on "Filter" significantly. On FB15k, TranSparse achieves the accuracy of 79.9% on *Hits@10*, which outperforms all the baseline methods. For the comparison of *Hits@10* of different kinds of relations, Table 5 shows the detailed results by mapping properties of relations on FB15k. In Table 5, TranSparse outperforms Trans(E, H ,R, and D) on all kinds of relations.

Analysis (1) In Table 4, TranSparse(share) outperforms Trans(E, H, R) and is near to TransD, which shows that our approach deals with the data heterogeneity well; (2) In Ta-

ble 4, TranSparse(separate) obtains better perfermance than TranSparse(share) in most cases. And in Table 5, for the typical unbalanced relations (1-to-N, N-to-1 and N-to-N), TranSparse(separate) outperforms TranSparse(share) a relatively large margin on most prediction accuracies. Hence, we can conclude that TranSparse(separate) can overcome the heterogeneity and imbalance of data simultaneously; and (3)Unstructured pattern also get better performance than structured pattern.

Conclusions and Future Work

We introduced a model named TranSparse that embed knowledge graphs into continues vector space with adaptive sparse transfer matrices for their completion. It considers the heterogeneity and imbalance of data. In addition, we provide two sparse patterns for transfer matrices and analyze their advantages and disadvantages. Extensive experiments show that TranSparse outperforms all baseline models on triplet classification and link prediction tasks. We will explore the best sparse patterns for transfer matrices in the future.

Acknowledgements

The work was supported by the Natural Science Foundation of China (No. 61533018), the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332). And this research work was also supported by Google through focused research awards program.

References

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. ACM.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, 127–135.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, 2787–2795.

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

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM (JACM)* 58(3):11.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

Guo, S.; Wang, Q.; Wang, B.; Wang, L.; and Guo, L. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of ACL*, 84–94.

He, S.; Liu, K.; Ji, G.; and Zhao, J. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. ACM, 2015*, 623–632.

Jenatton, R.; Roux, N. L.; Bordes, A.; and Obozinski, G. R. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, 3167–3175.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL*, 687–696.

Kennedy, R., and Examination II, W. P. 2013. Low-rank matrix completion. *unpublished (" http://www. seas. upenn. edu/kenry/")*.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, 2181–2187.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Nickel, M., and Tresp, V. 2013a. An analysis of tensor models for learning on structured data. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 272–287.

Nickel, M., and Tresp, V. 2013b. Tensor factorization for multi-relational learning. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 617–621.

Nickel, M.; Murphy, K.; Tresp, V.; and Gabrilovich, E. 2015. A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 809–816.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, 271–280. ACM.

Saad, Y. 2003. *Iterative methods for sparse linear systems*. Siam.

Socher, R.; Chen, D.; Manning, C. D.; and Ng, A. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, 926–934.

Sutskever, I.; Tenenbaum, J. B.; and Salakhutdinov, R. R. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, 1821–1828.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1112–1119. Citeseer.