TransGate: Knowledge Graph Embedding with Shared Gate Structure

Jun Yuan,[1,2,3] **Neng Gao,**[1,2] **Ji Xiang**[1,2*]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing, China
[3]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{yuanjun,gaoneng,xiangji}@iie.ac.cn

Abstract

Embedding knowledge graphs (KGs) into continuous vector space is an essential problem in knowledge extraction. Current models continue to improve embedding by focusing on discriminating relation-specific information from entities with increasingly complex feature engineering. We noted that they ignored the inherent relevance between relations and tried to learn unique discriminate parameter set for each relation. Thus, these models potentially suffer from high time complexity and large parameters, preventing them from efficiently applying on real-world KGs. In this paper, we follow the thought of parameter sharing to simultaneously learn more expressive features, reduce parameters and avoid complex feature engineering. Based on gate structure from LSTM, we propose a novel model TransGate and develop shared discriminate mechanism, resulting in almost same space complexity as indiscriminate models. Furthermore, to develop a more effective and scalable model, we reconstruct the gate with weight vectors making our method has comparative time complexity against indiscriminate model. We conduct extensive experiments on link prediction and triplets classification. Experiments show that TransGate not only outperforms state-of-art baselines, but also reduces parameters greatly. For example, TransGate outperforms ConvE and R-GCN with 6x and 17x fewer parameters, respectively. These results indicate that parameter sharing is a superior way to further optimize embedding and TransGate finds a better trade-off between complexity and expressivity.

Introduction

Knowledge graphs (KGs) such as WordNet (Miller 1995) and Freebase (Bollacker et al. 2008) have been widely adopted in various applications, like web search, Q&A, etc. KGs are large-scale multi-relational structures and usually organized in the form of triplets. Each triplet is denoted as (h, r, t), where h and t are head and tail entities, respectively, and r is the relation between h and r. For instance, (*Beijing*, *CapitalOf*, *China*) denotes the fact that the capital of *China* is *Beijing*.

Although existing KGs already have consisted of billions of triplets, but they are far from completeness. Thus, the knowledge graph completion (KGC) has emerged an important open research problem (Nickel, Tresp, and Kriegel 2011). For example, it is shown by (Dong et al. 2014) that more than 66% of the person entities are missing a birthplace in Freebase and DBLP. The aim of KGC task is to predict relations and determine specific-relation type between entities based on existing triplets. Due to the extremely large data size, an effective and scalable solution is crucial for KGC task (Liu, Wu, and Yang 2017).

Various KGC methods have been developed in recent years, and the most successful methods are representation learning based methods which encode KGs into low-dimensional vector spaces. Among these methods, TransE (Bordes et al. 2013) is a simple and effective model and regards every relation as translation between the heads and tails. We denote embedding vector with the same letters in boldface. When (h, r, t) holds, the embedding $\mathbf{h}$ is close to the embedding $\mathbf{t}$ by adding the embedding $\mathbf{r}$, that is $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$.

Indiscriminate models, like TransE, can scale to real-world KGs (Xu et al. 2017a), due to the limited number of parameters and low computational costs. However, indiscriminate models often learn less expressive features (Dettmers et al. 2018), leading to either low parameter efficiency or low accuracy. In fact, an entity may have multiple aspects which may be related to different relations (Lin et al. 2015). Therefore, relation-specific information discrimination of entities is the key to improving embeddings.

Many discriminate models (Lin et al. 2015; Ji et al. 2015; 2016) have been proposed recently and usually outperform indiscriminate models on public data sets. We note that current methods always assumed the independence between relations and tried to learn unique discriminate parameter set for each relation. Although there are always thousands of relations in a KG, but they are not independent from each other (Bordes et al. 2013). For example, among WordNet, the most frequently encoded relation is the hierarchical relation and each hierarchical relation has its inverse relation, $e.g.$ *hyponym/hypernym*.

As a result, current discriminate models potentially suffer from following three problems, which all prevent current models from efficiently applying on real-world KGs.

First, current discriminate models potentially suffer from large parameters. For instance, when applying TransD (Ji et al. 2015) on Freebase with an embedding size of 100, the pa-

*Corresponding author

rameter size will be more than 33GB and half parameters are discriminate parameters. Second, current discriminate models explore increasingly complex feature engineering to improve embedding, resulting in high time complexity. In reality, one nature of correlated relations is semantic sharing (Trouillon et al. 2016), which can be used to improve the discrimination. Third, due to the large parameter size and complex design, current models have to introduce more hyper parameters and adopt pre-training to prevent overfitting. Please refer to Table 1 and "Related Work" for details.

To optimize embedding and reduce parameters simultaneously, we follow the thought of parameter sharing and propose a novel method TransGate. Unlike previous methods tried to learn relation-specific discriminate parameter set, TransGate discriminates relation-specific information with only two shared gates for all relations. Hence, TransGate has almost same space complexity as TransE on real-world KGs. Furthermore, shared gates share statistical strength across different relations to optimize embedding without complex feature engineering.

We conduct extensive experiments on link prediction and triplets classification on large-scale public KGs, namely Freebase and WordNet. TransGate achieves better performance than ConvE (Dettmers et al. 2018) and R-GCN (Michael Sejr et al. 2018) with 6x and 17x fewer parameters, respectively. Unlike many of the related models that require pre-training, TransGate is a self-contained model and does not need any extra hyper parameter to prevent overfitting.

Our specific contributions are as follows:

- We identify the significance of inherent relevance between relations, which is largely overlooked by existing literature. We follow the thought of parameter sharing to learn more expressive features and reduce parameters at the same time.
- Based on gate structure from LSTM (Hochreiter and Schmidhuber 1997), we propose TransGate and develop shared discriminate mechanism to avoid large discriminate parameters.
- To develop a more effective and scalable model, we reconstruct standard gate with weight vectors. In this way, we reduce calculation brought by matrix-vector multiplication operations and decrease the time complexity to the same order as TransE.
- Experiments show that TransGate not only delivers significant improvements compared to state-of-art baselines, but also reduces parameters greatly. These results indicate that parameter sharing is a superior way to further optimize embedding and our method finds a better trade-off between complexity and expressivity.

Related Work

Indiscriminate Models

Indiscriminate models usually focus more on the scalability on real-world KGs. But they often have either low parameter efficiency or low accuracy (Dettmers et al. 2018).

TransE. As mentioned in Introduction section, TransE (Bordes et al. 2013) encodes entities and relations in the same space $\mathbb{R}^m$. Then it regards each relation as translation between the heads and tails and wants $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds. Hence, TransE assumes score function $f_r(h,t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_{L_1/L_2}$ is low if (h,r,t) holds, and high otherwise.

DistMult. DistMult (Yang et al. 2015) has the same time and space complexity as TransE. DistMult uses weighted element-wise dot product to define the score function $f_r(h,t) = \sum h_k r_k t_k$. Although DistMult has better overall performance than TransE, but it is unable to model asymmetric relations.

ComplEx. ComplEx (Trouillon et al. 2016) makes use of complex valued embeddings and Hermitian dot product to address the antisymmetric problem in DistMult. As mentioned in its paper, though ComplEx doubles the parameter size of TransE, but TransE and DistMult perform better on symmetry relations.

CombinE. CombinE (Tan, Zhao, and Wang 2017) considers triplets features from two aspects: relation $\mathbf{r}_p \approx \mathbf{h}_p + \mathbf{t}_p$ and entity $\mathbf{r}_m \approx \mathbf{h}_m - \mathbf{t}_m$. The score function is $f_r(h,t) = ||\mathbf{h}_p + \mathbf{t}_p - \mathbf{r}_p||^2_{L_1/L_2} + ||\mathbf{h}_m - \mathbf{t}_m - \mathbf{r}_m||^2_{L_1/L_2}$. CombinE doubles the parameter size of TransE, but does not yield significant boost in performance on link prediction.

Discriminate Models

Discriminate models focus more on precision. They usually contain two stages: relation-specific information discrimination and score computation.

TransH. TransH (Wang et al. 2014) maps entity embedding into relation hyperplanes to discriminate relation-specific information. The score function is defined as $f_r(h,t) = ||\mathbf{h} - \mathbf{w}_r^T\mathbf{h}\mathbf{w}_r + \mathbf{r} - (\mathbf{t} - \mathbf{w}_r^T\mathbf{t}\mathbf{w}_r)||_{L1/L2}$, where $\mathbf{w}_r$ is the normal vector of r's relation hyperplane.

TransR/CTransR. TransR (Lin et al. 2015) learns a mapping matrix $\mathbf{M}_r$ for each relation. It maps entity embedding into relation space to realize the discrimination. The score function is $f_r(h,t) = ||\mathbf{M}_r\mathbf{h} + \mathbf{r} - \mathbf{M}_r\mathbf{t}||_{L_1/L_2}$. CTransR is an extension of TransR. It clusters diverse head-tail entity pairs into groups and sets a relation vector for each group.

TranSparse. TranSparse (Ji et al. 2016) replaces transfer matrix in TransR by two sparse matrices for each relation. And it uses two sparse degree hyper parameters, θ_r^h and θ_r^t, to reduce parameters of mapping matrices.

TransD. TransD (Ji et al. 2015) constructs two mapping matrices dynamically for each triplet by setting projection vector for each entity and relation, that is $\mathbf{M}_{rh} = \mathbf{h}_p^\top\mathbf{r}_p + \mathbf{I}^{m\times n}$, $\mathbf{M}_{rt} = \mathbf{t}_p^\top\mathbf{r}_p + \mathbf{I}^{m\times n}$. The score function is $f_r(h,t) = ||\mathbf{M}_{rh}\mathbf{h} + \mathbf{r} - \mathbf{M}_{rt}\mathbf{t}||_{L1/L2}$. TransD obtains the state-of-art performance on triplets classification.

Neural Tensor Network (NTN). NTN (Socher et al. 2013) learns a 3-way tensor and two transfer matrices for each relation as discriminate parameter set. NTN constructs a neural network to discriminate information. The score function is defined as $f_r(h,r,t) = \mathbf{u}_r^T f(\mathbf{h}^\top\hat{\mathbf{M}}_r\mathbf{t} + \mathbf{M}_{r1}\mathbf{h} + \mathbf{M}_{r2}\mathbf{t} + \mathbf{b}_r)$, where $\hat{\mathbf{M}}_r \in \mathbb{R}^{m\times m\times s}$, and *f()* is *tanh* operation. The large parameters and high time complexity of NTN prevent it from applying on large-scale KGs (Shi and Weninger 2017).

To prevent overfitting, current discriminate models introduce more hyper parameters and require pre-trained data from prerequisite models. In addition, they often explore linear discrimination to avoid high time complexity of non-linear operation. Table 1 lists the details of several models. TransGate achieves non-linear discrimination with relative low time and space complexity by shared gates.

Other Models

A number of works attempt to improve knowledge graph embedding in different ways. Some models explore different loss function to improve embeddings. Zhou et al. (Zhou et al. 2017) propose a limit-based scoring loss function to provide lower scoring of a golden triplet. ManifoldE (Xiao, Huang, and Zhu 2016) proposes a manifold-based loss function to achieve precise link prediction. Additionally, some additional information has been used to improve embedding. The NLFeat model (Toutanova and Chen 2015) is a log-linear model using simple node and link features. Jointly (Xu et al. 2017b) uses a gating mechanism to integrate representations of structure and text information of entities. RUGE (Guo et al. 2018) utilizes the soft-rule between triplets to enhance ComplEX. But the extraction of additional information is always the bottleneck.

Many researches attempt to introduce some novel techniques of deep learning into knowledge graph embedding. KBGAN (Cai and Wang 2018) introduces GAN to boost several embedding models. ProjE (Shi and Weninger 2017) uses a learnable combination operator to combine embeddings and feeds combined embeddings in a multi-class classifier to handle complex relations. Large candidate entity number N_c will cause high time complexity, so ProjE has to use candidate sampling to reduce N_c. R-GCN (Michael Sejr et al. 2018) and ConvE (Dettmers et al. 2018) introduces multi-layer convolution network in knowledge graph embedding. ConvKB (Dai et al. 2018) designs a simple convolutional model and achieves the state-of-art results on link prediction. ConvKB concatenates head, tail and relation vectors into a $3 \times m$ matrix, then feeds it into a convolution network. The score function is defined as $f_r(h,t) = \text{concat}(g([\mathbf{h},\mathbf{r},\mathbf{t}] * \mathbf{\Omega}))\mathbf{w}$, where $*$ means convolution operation; $\mathbf{\Omega} \in \mathbb{R}^{\tau \times 1 \times 3}$ is the set of filters; τ is the number of filters; $\mathbf{w} \in \mathbb{R}^{\tau m \times 1}$. To learn expressive features, the magnitude of τ is same as dimension m in above three convolution models.

TransGate is a shallow model without using any additional information. This make it scale to real-world KGs easier.

Embedding with Shared Gate Structure

In this work, we follow the thought of parameter sharing and propose an effective and scalable method named TransGate. In this section we firstly introduce gate structure of LSTM. Then we describe the TransGate architecture, along with the training method. Last but not least, we make complexity analysis on TransGate compared with some baselines.

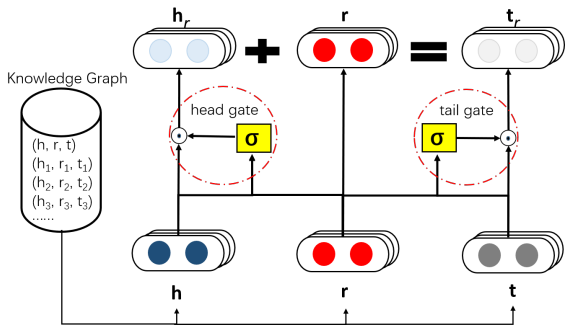

Figure 1: TransGate Architecture. TransGate discriminates relation-specific information with only two shared gates for all relations. Shared gates can share the statistical strength across different relations to optimize embedding.

Gate structure

Gate structure is the core mechanism of LSTM (Hochreiter and Schmidhuber 1997), which has been found extremely successful in many applications (Józefowicz, Zaremba, and Sutskever 2015). Gate is a way to optionally let information through. As marked in the red dashed circle in Figure 1, a gate is composed of a layer with sigmoid activation function and a Hadamard product operation.

The standard gate uses fully connected layer, which is shown as Equation 1. The sigmoid operation is applied element-wise and [,] means concatenate operation.

$$\mathbf{f} = \sigma(\mathbf{b} + \mathbf{W} \cdot [\mathbf{x}, \mathbf{y}]), \tag{1}$$

where $\mathbf{x}$ is the current input vector and $\mathbf{y}$ is the representation vector of the context. $\mathbf{b}$, $\mathbf{W}$ are respectively bias and weight matrix for the gate structure.

Sigmoid operation sets every element of $\mathbf{f}$ between 0 and 1, describing how much information should be let through. Afterwards, a gate uses Hadamard product to filter information as Equation 2

$$\mathbf{s}_f = \mathbf{s} \odot \mathbf{f}, \tag{2}$$

where $\mathbf{s}$ is the vector that we want to filter, $\mathbf{s}_f$ is the filtered information vector, and $\odot$ means the Hadamard product.

All the parameters of gate need to be learned during training. Therefore, theoretically, a gate can learn to adaptively and nonlinearly filter information based on the input with fixed number of parameters. That is to say, we can train a gate making the filtered vector $\mathbf{s}_f$ related to various $\mathbf{x}, \mathbf{y}$ without extra parameters. In practice, this great ability of gate structure has been shown in many researches (Józefowicz, Zaremba, and Sutskever 2015).

TransGate Architecture

The main insight in development of TransGate is that we assume that relation-specific information of entity will reflect the relevance between relations too. In other words, there should exist some common features in different relation-specific embedding vectors of related relations. Hence, we develop a shared discriminate mechanism for all relations based on gate structure. Shared gate shares statistical strength across different relations to optimize embedding.

The framework of TransGate is shown as Figure 1 and the detailed descriptions are as follow:

1. We embed every entity and relation into continues vector with same dimension.
2. To handle "one-relation-circle" structures (Zhang 2017) [1], TransGate sets two shared gates for head and tail entities, respectively.
3. We input both entity embedding and relation embedding into the sigmoid layer of gate, letting the discrimination determined by entity and relation together. [2]
4. Then we realize non-linear and adaptive relation-specific discrimination through multiplying sigmoid layer output and entity embedding element-wise.
5. Last, we build translation between discriminated information of heads and tails. Based on the score, we can determine whether the triplet is valid.

Next, we describe TransGate using standard gate with fully connected layer, or **TransGate(fc)**. Considering a training set S of triplets (h, r, t), each triplet is composed of two entities $h, t \in E$ and relations $r \in R$, where E and R are the set of entities and relations, respectively. Embeddings are set as $\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^m$. And the parameters of two gates are denoted as $\mathbf{W}_h, \mathbf{W}_t \in \mathbb{R}^{m \times 2m}$ and $\mathbf{b}_h, \mathbf{b}_t \in \mathbb{R}^m$. We define the relation-specific entity embedding vectors as

$$\mathbf{h}_r = \mathbf{h} \odot \sigma(\mathbf{W}_h \cdot [\mathbf{h}, \mathbf{r}] + \mathbf{b}_h), \tag{3}$$

$$\mathbf{t}_r = \mathbf{t} \odot \sigma(\mathbf{W}_t \cdot [\mathbf{t}, \mathbf{r}] + \mathbf{b}_t). \tag{4}$$

The fully connected layer interacts different embedding dimensions and can make precise discrimination.

To develop a more effective and scalable model, we reconstruct the standard gate in our method. Theoretically, every dimension should be independent from each other in a well-trained embedding model. Hence, we replace matrix in gate with two weight vectors and propose **TransGate(wv)**. Correspondingly, the relation-specific entity embedding vectors are define as

$$\mathbf{h}_r = \mathbf{h} \odot \sigma(\mathbf{V}_h \odot \mathbf{h} + \mathbf{V}_{r_h} \odot \mathbf{r} + \mathbf{b}_h), \tag{5}$$

$$\mathbf{t}_r = \mathbf{t} \odot \sigma(\mathbf{V}_t \odot \mathbf{t} + \mathbf{V}_{r_t} \odot \mathbf{r} + \mathbf{b}_t), \tag{6}$$

where weight vectors $\mathbf{V}_h, \mathbf{V}_t, \mathbf{V}_{r_h}, \mathbf{V}_{r_t} \in \mathbb{R}^m$. In this way, we can reduce the calculation by avoiding matrix-vector multiplication operations.

After discrimination, we build translation on relation-specific information and the score function is correspondingly defined as

$$f_r(h, t) = ||\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r||_{L_1/L_2}, \tag{7}$$

The score is expected to be lower for a golden triplet and higher for an incorrect triplet.

[1] One-relation-circle means that some entities form a circle with the same relation in a KG. One-relation-circle will make the corresponding relation vector **r** approximate 0.

[2] The discriminate operation should be an interactive process between an entity and a relation, because a relation may link to different types entities (Ji et al. 2015).

In practice, we enforce constraints on the norms of the embeddings and discriminated embeddings. That is to say, $\forall h,\ t,\ r$, we have $||\mathbf{h}||_2 = 1, ||\mathbf{r}||_2 = 1, ||\mathbf{t}||_2 = 1, ||\mathbf{h}_r||_2 = 1$ and $||\mathbf{t}_r|| = 1$. We make no constraint on weight matrices, weight vectors and biases of gates.

TransGates can discriminate relation-specific information for different entities and relations with fixed and small number of parameters. We note that when head gate and tail gate let all information through, TransGate will degenerate to TransE. This suggests that TransGate embraces TransE (Bordes et al. 2013), being a more general knowledge graph embedding framework.

Training

To learn such embeddings, we minimize a common used margin-based ranking criterion over the training set:

$$\mathcal{L} = \sum_{(h,r,t) \in S} \sum_{(h',r,t') \in S'} [\gamma + f_r(h, t) - f_r(h', t')]_+, \tag{8}$$

where $[x]_+ \triangleq \max(0, x)$, $\gamma > 0$ is a margin hyper parameter, the corrupted triplet set S' is composed of training triplets with the entities or relations replaced randomly. That is

$$\begin{aligned} S' = &\{(h', r, t) | h' \in E\} \cup \{(h, r, t') | t' \in E\} \\ &\cup \{(h, r', t) | r' \in R\} \end{aligned} \tag{9}$$

The loss function 8 is used to encourage discrimination between training triplets and corrupted triplets by favoring lower scores for training triplets than for corrupted ones.

We initialize the embeddings, weight matrices and weight vectors of gates through sampling from a truncated standard normal distribution. And we initialize the biases as vectors that all elements are 1. The training process of TransGate is carried out using Adam optimizer (Kingma and Ba 2015) with constant learning rate. The training process is stopped based on model's performance on the validation set.

Complexity Analysis

In Table 1, we compare the embedding parameter number, discriminate parameter number, hyper parameter number, time complexity of each iteration and pre-training methods of various baselines with our model. N_e, N_r represent the number of entities, relations, respectively. N_c represents the number of candidate entities. 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 hidden nodes' number of a neural network and s is the number of slice of a tensor. $\hat{\theta}$ denotes the average sparse degree of all transfer matrices. τ is the number of convolutional kernels.

From Table 1 we can see that:

1. The space complexity of TransGate is almost the same as TransE on real-world KGs. The discriminate parameters brought by the gates can be ignored compare to embedding parameters. Because $m \ll N_r \ll N_e$ among existing KGs, so $\frac{parameters\ of\ standard\ gate}{parameters\ of\ Embedding\ parameters} = \frac{4m+2}{N_e+N_r} \approx 0$.

Table 1: Complexities (the number of embedding parameters, discriminate parameters and hyper parameters, time complexity in an epoch, and pre-training method) of several embedding models.

Model	#Embedding Parameters	#Discriminate Parameters	#Hyper Parameters	Time complexity	Pre-training Method
SLM (Socher et al. 2013)	$O(N_e m + N_r n)(m = n)$	$O(N_r(2k + 2nk))$	2	$O(mk)$	Word Embedding
NTN (Socher et al. 2013)	$O(N_e m + N_r n)(m = n)$	$O(N_r(n^2 s + 2nk + 2k))$	2	$O(m^2 s + mk)$	Word Embedding
TransE (Bordes et al. 2013)	$O(N_e m + N_r n)(m = n)$	None	2	$O(m)$	None
TransH (Wang et al. 2014)	$O(N_e m + N_r n)(m = n)$	$O(N_r n)$	4	$O(m)$	TransE
DistMult (Yang et al. 2015)	$O(N_e m + N_r n)(m = n)$	None	2	$O(m)$	None
TransR (Lin et al. 2015)	$O(N_e m + N_r n)$	$O(N_r mn)$	3	$O(mn)$	TransE
CTransR (Lin et al. 2015)	$O(N_e m + N_r dn)$	$O(N_r mn)$	4	$O(mn)$	TransR
TransD (Ji et al. 2015)	$O(N_e m + N_r n)$	$O(N_e m + N_r n)$	3	$O(m)$	TransE
TranSparse (Ji et al. 2016)	$O(N_e m + N_r n)$	$O(2N_r(1 - \hat{\theta})mn)(0 \leq \hat{\theta} \leq 1)$	5	$O(2(1 - \hat{\theta})mn)(0 \leq \hat{\theta} \leq 1)$	TransE
ComplEx (Trouillon et al. 2016)	$O(2N_e m + 2N_r n)(m = n)$	None	2	$O(m)$	None
CombinE (Tan, Zhao, and Wang 2017)	$O(2N_e m + 2N_r n)$(m=n)	None	2	$O(2m)$	None
ProjE (Shi and Weninger 2017)	$O(N_e m + N_r n + 5m)$(m=n)	None	2	$O(N_e m + 2m)$	None
ConvKB (Dai et al. 2018)	$O(N_e m + N_r n + (\tau + 3)m)$(m=n)	None	3	$O(\tau m)$	None
TransGate(fc) (this paper)	$O(N_e m + N_r n)(m = n)$	$O(4m^2 + 2m)$	2	$O(m^2)$	None
TransGate(wv) (this paper)	$O(N_e m + N_r n)(m = n)$	$O(4m + 2n)(m = n)$	2	$O(m)$	None

2. TransGate(wv) introduces non-linear discrimination with similar time complexity to TransE and TransD. Although time complexity of TransGate(fc) is higher than TransGate(wv), but it is still similar to TransR and ConvKB.
3. Our methods introduce no more hyper parameter than TransE. Besides, TransGates do not need any hyper parameter or pre-training to prevent overfitting. This makes TransGates can be trained easier.
4. Due to the low time complexity and less parameters, TransGates can easily be applied to real-world KGs, using less computing and memory resources.

Experiments

In this section, we empirically evaluate our proposed models on two key tasks: link prediction (Bordes et al. 2013) and triplets classification (Socher et al. 2013). We demonstrate that TransGates outperform baselines and deliver significant improvements on multiple benchmark data sets. We also investigate the parameter efficiency of our methods. Results show that parameter sharing is a superior way to further optimize embedding and TransGates find a better trade-off between complexity and expressivity.

Data Sets

Link prediction and triplets classification are implemented on two large-scale knowledge graphs: WordNet (Miller 1995) and Freebase (Bollacker et al. 2008). WordNet provides semantic knowledge of words. Entities in WordNet are synonyms which express distinct concepts. Relations in WordNet are conceptual-semantic and lexical relations. In this paper, we employ two data sets from WordNet: WN11 (Socher et al. 2013) and WN18RR (Dettmers et al. 2018). Freebase provides general facts of the world. For example, the triplet (*Anthony Asquith, location, London*) builds a relation of *location* between the name entity *Anthony Asquith* and the city entity *London*. In this paper, we employ three data sets from Freebase: FB15K (Bordes et al. 2013), FB13 (Socher et al. 2013) and FB15K-237 (Toutanova and Chen 2015). Table 3 lists statistics of the five data sets.

WN18RR and FB15K-237 are correspondingly subsets of two common data sets WN18 (Bordes et al. 2013) and FB15K. It is firstly discussed by (Toutanova and Chen 2015) that WN18 and FB15K suffer from test leakage through inverse relations, $i.e.$ many test triplets can be obtained simply by inverting triplets in the training set. To address this issue, (Toutanova and Chen 2015) generated FB15K-237 by removing redundant relations in FB15K and greatly reducing the number of relations. Likewise, (Dettmers et al. 2018) removed reversing relations in WN18. As a consequence, the difficulty of reasoning on these two data sets is increased dramatically. Since FB15K is still widely used, we also include results on this data set.

Link Prediction

Link prediction aims to predict the missing h or t for a triplet (h, r, t). In this task, the model is asked to rank a set of candidate entities from the KG, instead of giving one best result. For each test triplet (h, r, t), we replace the head/tail entity by all possible candidates in the KG, and rank these entities in ascending order of scores calculated by score function f_r.

We follow the evaluation protocol in (Bordes et al. 2013) to report filtered results. Because a corrupted triplet, generated in the aforementioned process of removal and replacement, may also exist in KG, and it should be considered as correct. In other words, we filtered out the correct triplets from corrupted triplets which exist in the training set.

We report three common measures as our evaluation metrics: the average rank of all correct entities (*Mean Rank*), the mean reciprocal rank of all correct entities (*MRR*), and the proportion of correct entities ranked in top K (*Hits@K*). A good link predictor should achieve lower *Mean Rank*, higher *MRR*, and higher *Hits@K*.

In this task, we use three data sets: WN18RR, FB15K and FB15K-237. Since the data sets are same, we directly copy experimental results of several baselines from (Dettmers et al. 2018; Cai and Wang 2018; Dai et al. 2018). We select the hyper parameters of TransGate via grid search according to the *Hits@10* on the validation set. In training, we use the same configurations for both TransGate(fc) and TransGate(wv). For three data sets, we traverse all the training triplets for at most 1000 rounds. We search the learning rate α for Adam among $\{0.01, 0.1, 0.5\}$, the margin γ among $\{1, 2, 3, 4, 5, 6\}$, the embedding dimension m among $\{32, 50, 100, 200\}$, and the batch size B among $\{120, 480, 1440\}$.

Table 2: Experimental Results of Link Prediction.

Model	WN18RR				FB15K				FB15K-237			
	MRR	**Mean Rank**	**Hits@10(%)**	**Hits@1(%)**	**MRR**	**Mean Rank**	**Hits@10(%)**	**Hits@1(%)**	**MRR**	**Mean Rank**	**Hits@10(%)**	**Hits@1(%)**
TransE (Bordes et al. 2013)	0.226	3384	50.1	-	0.220	125	47.1	23.1	0.294	347	46.5	14.7
DistMult (Yang et al. 2015)	0.43	5110	49.0	39.0	0.654	97	82.4	54.6	0.241	254	41.9	15.5
TransD (Ji et al. 2015)	-	-	42.8	-	0.252	67	77.3	23.4	-	-	45.3	-
CombinE (Tan, Zhao, and Wang 2017)	-	-	-	-	0.283	-	85.2	55.4	-	-	-	-
ComplEX (Trouillon et al. 2016)	**0.44**	5261	51.0	**41.0**	0.692	-	84.0	59.9	0.247	339	42.8	15.8
ANALOGY (Liu, Wu, and Yang 2017)	-	-	-	-	0.725	-	85.4	64.6	-	-	-	-
KB-LRN (Garciaduran and Niepert 2017)	-	-	-	-	0.794	44	87.5	74.8	0.309	209	49.3	21.9
NLFeat (Toutanova and Chen 2015)	-	-	-	-	0.822	-	87.0	-	0.249	-	41.7	-
RUGE (Guo et al. 2018)	-	-	-	-	0.768	-	86.5	70.3	-	-	-	-
KBGAN (Cai and Wang 2018)	0.213	-	48.1	-	-	-	-	-	0.278	-	45.8	-
R-GCN (Michael Sejr et al. 2018)	-	-	-	-	0.696	-	0.842	60.1	0.248	-	41.7	15.3
ConvE (Dettmers et al. 2018)	0.43	4187	52.0	40.0	0.657	51	83.1	55.8	0.325	244	50.1	23.7
ConvKB (Dai et al. 2018)	0.248	**2554**	**52.5**	-	-	-	-	-	0.396	257	51.7	-
TransGate(fc)	0.409	3420	51.0	39.1	**0.832**	**33**	**91.4**	**75.5**	**0.404**	177	**58.1**	**25.1**
TransGare(wv)	0.396	3227	51.4	38.7	**0.831**	**32**	**91.4**	**75.6**	**0.396**	**144**	54.2	24.2

Table 3: Statistics of data sets

Dataset	#Rel	#Ent	#Train	#Valid	#Test
WN11	11	38,696	112,581	2,609	10,544
WN18RR	11	40,943	86,835	3,034	3,134
FB13	13	75,043	316,232	5,908	2,3733
FB15K	1,345	14,951	483,142	50,000	59,071
FB15K-237	237	14,541	272,115	17,535	20,466

The best configurations are as follow: on WN18RR, $\gamma = 5$, $\alpha = 0.1$, $m = 200$, $B = 120$ and taking L_1 dissimilarity; on FB15K, $\gamma = 1$, $\alpha = 0.1$, $m = 200$, $B = 480$ and taking L_1 dissimilarity; on FB15K-237, $\gamma = 4$, $\alpha = 0.5$, $m = 200$, $B = 480$ and taking L_1 dissimilarity.

From Table 2, we observe that: (1) TransGate(fc) and TransGate(wv) outperform all baselines on FB15K and FB15K-237 at every metric. (2) The performance of our methods on WN18RR are similar to TransE and near to the best model. (3) In terms of *Hits10*, TransGate(fc) achieves the best result with 58.1% on FB15K-237, while TransGate(wv) achieves the second best result with 54.2%. On FB15K our methods both achieve the best accuracy at 91.4%. (4) In terms of *Mean Rank*, TransGate(wv) reduces the metric greatly by 31.1% on FB15K-237 and 27.3% on FB15K. TransGate(fc) also reduces the metric by 15.3% on FB15K-237 and 25% on FB15K. (5) In terms of *Hits1*, TransGate(fc) and TransGate(wv) outperform all baselines on FB15K and FB15K-237. This indicates the great ability of TransGates on precise link prediction.

WN18RR (Dettmers et al. 2018) has been removed reversing relations and destroyed the inherent structure of WordNet, resulting in low relevance between relations. Thus, TransGates perform similar to TransE on WN18RR. However, our methods still achieve state-of-art results on FB15K and FB15K-237. It should be noted that parameter amount of gate structure is fixed and small, only 40.2K in TransGate(fc) and 0.6K in TransGate(wv). These results show the appropriateness of sharing discriminate parameters and the great ability of gate structure.

Triplets Classification

Triplets classification is a binary classification task, aiming to judge whether a given triplet (h, r, t) is correct or not. In this paper, we use three data sets 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 triplets. We construct negative triplets for FB15K following the same setting used for FB13 (Socher et al. 2013).

For triplets classification, we set a relation-specific threshold δ_r. δ_r is optimized by maximizing classification accuracies on the validation set. For a triplet (h, r, t), if the score obtained by f_r is below δ_r, the triplet will be classified as positive, otherwise negative.

For WN11 and FB13, we compare TransGate with baselines reported in (Ji et al. 2016) and (Zhou et al. 2017). Since FB15K is generated by ourselves, we use the codes provided by (Lin et al. 2015) to re-evaluate the data set instead of reporting the results of (Zhou et al. 2017) directly.

In training, we use the same configurations for both TransGate(fc) and TransGate(wv). We use learning rate α for Adam among {0.001, 0.01, 0.1}, the margin γ among {2, 4, 6, 10}, the embedding dimension m among {20, 50, 100}, and the batch size B among {120, 480, 1440}. The optimal configurations are determined by the validation set. On WN111, the best configurations are: $\gamma = 10$, $\alpha = 0.01$, $m = 100$, $B = 120$ and taking L_1 dissimilarity. On FB13, the best configurations are: $\gamma = 6$, $\alpha = 0.001$, $m = 100$, $B = 1440$ and taking L_1 dissimilarity. On FB15K, the best configurations are: $\gamma = 6$, $\alpha = 0.1$, $m = 100$, $B = 480$ and taking L_1 dissimilarity. For three data sets, we traverse training triplets for 400 rounds at most.

Table 4 shows the detailed evaluation results of triplets classification. From Table 4, we observe that: (1) On WN11, our methods outperform all baseline models, which obtains the accuracy of 87.3%. (2) On FB13, the accuracy of our methods are near to the best accuracy of TransD and higher than the other baselines. (3) On FB15K, TransGate(fc) achieves best result by 89.5%, while TransGate(wv) also achieves 89.3%. (4) TransGate(fc) performs better than TransGate(wv) on this task.

The results of TransGates indicate the appropriate design of the model. TransGates are able to better handle not only the sparse graph FB15K, but also the dense graph WN11. This indicates the good generalization of our methods. Additionally, we implemented our methods and TransE in TensorFlow (Abadi et al. 2016). Note that all models were run on a standard hardware of Inter(R) Core(TM) i7 2.6GHz + GeForce GTX 960M, with the same mini-batch size. The

training time of TransE is 4.97min, while TransGate(wv) is 5.11min and TransGate(fc) is 9.69min. In contrast, as shown in (Lin et al. 2015), the training time of TransH and TransR are about 6 and 60 times more than TransE, respectively. These results provide evidence toward that time complexity of TransGate(wv) is similar to TransE in another aspect.

Table 4: Experimental Results of Triplets Classification(%)

Models	WN11	FB13	FB15K
SE (Bordes et al. 2011)	53.0	75.2	-
SME(bilinear) (Bordes, Glorot, and Weston 2012)	70.0	63.7	-
LFM (Jenatton et al. 2012)	73.8	84.4	-
SLM (Socher et al. 2013)	69.9	85.3	-
NTN (Socher et al. 2013)	70.4	87.1	68.3
TransE (Bordes et al. 2013)	75.9	70.9	78.0
TransH (Wang et al. 2014)	77.7	76.5	74.9
TransR (Lin et al. 2015)	85.5	74.7	81.7
CTransR (Lin et al. 2015)	85.7	-	83.9
TransD (Ji et al. 2015)	85.6	**89.1**	85.3
TranSparse (Ji et al. 2016)	86.8	86.5	87.2
TransE-RS (Zhou et al. 2017)	85.2	82.8	82.0
TransH-RS (Zhou et al. 2017)	86.3	82.1	83.0
TransGate(fc)	**87.3**	88.8	**89.5**
TransGate(wv)	**87.3**	88.1	89.3

Parameter efficiency of TransGate

For in-depth investigation on expressivity, we look into the performance on link prediction of our methods with various parameter size on FB15K-237. Table 5 shows the detailed results along with several baselines. γ is set to 4 during all TransGate training over different embedding size. From Table 5, we observe that

1. TransGates learn more expressive features than compared models, including deep models. TransGate(fc) with only 0.74M parameters and TransGate(wv) with only 1.48M parameters still surpass the best result at *Hits@10*. By contrast, multi-layer model R-GCN (Michael Sejr et al. 2018) achieves only 41.7%, and another deep model ConvE (Dettmers et al. 2018) yields 49.0%.
2. Our methods are highly parameter efficient and will grow much slower than all baselines. Since TransGates can outperform all baselines with smaller embedding size and fixed number of additional parameters. TransGate(fc) achieves better performance at *Hits@10* than ConvKB, ConvE, R-GCN with 1.5x, 6x and 17x fewer parameters. In other words, TransGate(fc) can outperform all baselines with only 7.9GB parameters for the entirety of Freebase, while R-GCN will be more than 82GB and ConvE will be more than 56GB.
3. Moreover, the time complexity of TransGate(fc) is similar to ConvE and ConvKB, $i.e.$ $O(m^2)$. TransGate(wv) improves the *Hits@10* from TransE by near 10% with same time complexity and half parameters of TransE. These results indicate that TransGate(wv) is more effective and scalable than TransE on large-scale KGs.

In general, experimental results indicate that parameter sharing is a superior way to further improve embedding and TransGates find a better trade-off between complexity and expressivity. Because TransGates not only reduce parameters greatly, but also deliver a significant improvement. TransGate(fc) and TransGate(wv) have their own advantages: TransGate(fc) is more parameter efficient and TransGate(wv) has lower time complexity. Researchers can use their advantages to tackle various problems.

Table 5: Prameter Scaling of Several Models

Model	Parameter size	Embedding size	MRR	Hits@10(%)
TransE	1.48M	100	0.22	39.8
DistMult	1.89M	128	0.23	41.0
DistMult	0.95M	64	0.22	39.0
R-GCN	8.28M	200	0.249	41.7
ConvE	5.05M	200	0.32	49.0
ConvE	1.89M	96	0.32	49.0
ConvE	0.95M	54	0.30	46.0
ConvKB	1.97M	100	0.396	51.7
TransGate(fc)	3.03M	200	0.404	**58.1**
TransGate(fc)	1.49M	100	0.363	**53.0**
TransGate(fc)	0.74M	50	0.328	**51.7**
TransGate(fc)	0.47M	32	0.263	41.5
TransGate(wv)	2.95M	200	0.396	**54.2**
TransGate(wv)	1.48M	100	0.347	**52.9**
TransGate(wv)	0.74M	50	0.297	49.5
TransGate(wv)	0.47M	32	0.246	39.6

Conclusion and Future Work

In this paper, we focus on embedding knowledge graphs into continuous vector spaces for knowledge graph completion. We find that previous methods ignore inherent relevance between relations and potentially suffer from huge parameters, preventing them from applying on real-world KGs efficiently. We follow parameter sharing to optimize embedding and reduce parameters simultaneously. We develop shared discriminate mechanism based on gate structure and propose TransGate. To avoid matrix-vector multiplication operations, we reconstruct standard gate by using the weight vectors and propose TransGate(wv). In this way, we decrease the time complexity to the same order as TransE. Extensive experiments on the tasks of link prediction and triplets classification show that parameter sharing is a superior way to further improve embedding and the great ability of gate structure in knowledge graph embedding. Because TransGates not only reduce parameters greatly, but also deliver a significant improvement compared to state-of-art models. These results also show that our methods are better trade-off between complexity and expressivity.

In the future, we will explore following three research directions: (1) We will explore the better parameter sharing mechanism and try to solve more problems in knowledge graph embedding. (2) We will adopt more sophisticated techniques to enhance TranGate, such as incorporating additional information, different loss functions and self-attention, etc. (3) Gate is a parameter efficient and fast operator, which can be composed into deep networks. We will extend TransGate to multi-layer model in the future.

Acknowledgement

This work is supported by the National Key Research and Development Program of China, and National Natural Science Foundation of China (No. U163620068).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; Kudlur, M.; Levenberg, J.; Monga, R.; Moore, S.; Murray, D. G.; Steiner, B.; Tucker, P. A.; Vasudevan, V.; Warden, P.; Wicke, M.; Yu, Y.; and Zheng, X. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of USENIX*, 265–283.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase:a collaboratively created graph database for structuring human knowledge. In *Proceddings of SIGMOD*, 1247–1250.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of AAAI*.

Bordes, A.; Usunier, N.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of NIPS*, 2787–2795.

Bordes, A.; Glorot, X.; and Weston, J. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *Proceedings of AISTATS*.

Cai, L., and Wang, W. Y. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of NAACL*.

Dai, Q. N.; Tu, D. N.; Nguyen, D. Q.; and Phung, D. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL*.

Dettmers, T.; Pasquale, M.; Pontus, S.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*, 1811–1818.

Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; and Zhang, W. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of SIGKDD*, 601–610.

Garciaduran, A., and Niepert, M. 2017. Kblrn : End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *Proceedings of UAI*.

Guo, S.; Wang, Q.; Wang, L.; Wang, B.; and Guo, L. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of AAAI*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Jenatton, R.; Roux, N. L.; Bordes, A.; and Obozinski, G. 2012. A latent factor model for highly multi-relational data. In *Proceedings of NIPS*, 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.

Ji, G.; Liu, K.; He, S.; and Zhao, J. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of AAAI*, 985–991.

Józefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of ICML*, volume 37, 2342–2350.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

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

Liu, H.; Wu, Y.; and Yang, Y. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of ICML*.

Michael Sejr, S.; Thomas N, K.; Peter, B.; Rianne, v. d. B.; Ivan, T.; and Max, W. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Miller, G. A. 1995. Wordnet: a lexical database for english. *COMMUNICATIONS OF THE ACM* 38(11):39–41.

Nickel, M.; Tresp, V.; and Kriegel, H. P. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, 809–816.

Shi, B., and Weninger, T. 2017. Proje: Embedding projection for knowledge graph completion. In *Proceedings of AAAI*.

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

Tan, Z.; Zhao, X.; and Wang, W. 2017. Representation learning of large-scale knowledge graphs via entity feature combinations. In *Proceedings of CIKM*, 1777–1786.

Toutanova, K., and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of CVCS*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *Proceedings of ICML*, volume 48, 2071–2080.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, 1112–1119.

Xiao, H.; Huang, M.; and Zhu, X. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of IJCAI*.

Xu, H.; Yankai, L.; Ruobing, X.; Zhiyuan, L.; and Maosong, S. 2017a. Openke: An open-source framework for knowledge embedding. http://openke.thunlp.org/home.

Xu, J.; Qiu, X.; Chen, K.; and Huang, X. 2017b. Knowledge graph representation with jointly structural and textual encoding. In *Proceedings of IJCAI*, 1318–1324.

Yang, B.; Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of ICLR*.

Zhang, W. 2017. Knowledge graph embedding with diversity of structures. In *Proceedings of WWW*, 747–753.

Zhou, X.; Zhu, Q.; Liu, P.; and Guo, L. 2017. Learning knowledge embeddings by combining limit-based scoring loss. In *Proceedings of CIKM*, 1009–1018.