ParamE: Regarding Neural Network Parameters as Relation Embeddings for Knowledge Graph Completion

Feihu Che,[1,2] **Dawei Zhang,**[1] **Jianhua Tao,**[1,2,3] **Mingyue Niu,**[1,2] **Bocheng Zhao**[1,2]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{chefeihu2017, niumingyue2017, zhaobocheng2015}@ia.ac.cn, {dawei.zhang, jhtao}@nlpr.ia.ac.cn

Abstract

We study the task of learning entity and relation embeddings in knowledge graphs for predicting missing links. Previous translational models on link prediction make use of translational properties but lack enough expressiveness, while the convolution neural network based model (ConvE) takes advantage of the great nonlinearity fitting ability of neural networks but overlooks translational properties. In this paper, we propose a new knowledge graph embedding model called ParamE which can utilize the two advantages together. In ParamE, head entity embeddings, relation embeddings and tail entity embeddings are regarded as the input, parameters and output of a neural network respectively. Since parameters in networks are effective in converting input to output, taking neural network parameters as relation embeddings makes ParamE much more expressive and translational. In addition, the entity and relation embeddings in ParamE are from feature space and parameter space respectively, which is in line with the essence that entities and relations are supposed to be mapped into two different spaces. We evaluate the performances of ParamE on standard FB15k-237 and WN18RR datasets, and experiments show ParamE can significantly outperform existing state-of-the-art models, such as ConvE, SACN, RotatE and D4-STE/Gumbel.

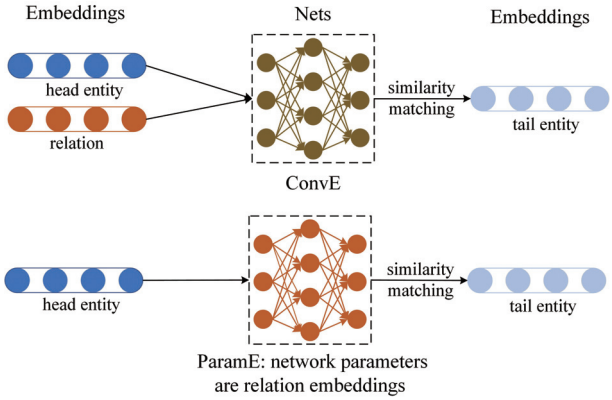

Figure 1: A comparsion of ConvE and ParamE. In some neural network based models such as ConvE, head entity embeddings and relation embeddings are fed together into neural networks, which ignores translational properties of relations. In the ParamE model, we regard neural network parameters as relation embeddings, such that the ParamE can make use of translational properties.

Introduction

Knowledge graphs(KGs) are composed of factual triplets, where each triplet represents a relation between a head entity and a tail entity. For example, a triple (*Chatou, isLocatedIn, France*) represents a relation (*isLocatedIn*) between a head entity (*Chatou*) and a tail entity(*France*). There are several real-world knowledge graphs, such as Freebase (Bollacker et al. 2008), Yago3 (Mahdisoltani, Biega, and Suchanek 2013), and WordNet (Miller 1995). They enable many downstream NLP tasks including information retrieval (Xiong, Power, and Callan 2017) and dialogue management (He et al. 2017). Due to the complexity of the real world, knowledge graphs lack of many triplets (Das et al. 2017). Link prediction is the task of predicting missing facts based on the existing ones (Nguyen 2017).

One of the effective ways for link prediction is knowledge graph embedding (Wang et al. 2017). The key idea is to map entities and relations in KGs into vector spaces, such that the score function of entities and relations for ground truth can distinguish from false facts efficiently (Yang et al. 2014).

In the existing link prediction models, there are two representative kinds of models: translational models and the convolutional neural network (CNN) based model (ConvE) (Dettmers et al. 2018). Translational models take relations as transfer functions which convert head entities to tail entities, such as TransE (Bordes et al. 2013), DistMul (Yang et al. 2014), ComplEx (Trouillon et al. 2016), Rotate (Sun et al. 2019). They make full use of the translational properties of relations but they only utilize simple operations like addition and multiplication, which limits the translational ability of relations (Shang et al. 2019). ConvE takes advantage of nonlinearity modeling ability of neural networks and has stronger expresiveness ability. However,

since ConvE feeds the neural network with entities and relations together, it ignores the translational properties of relations in KGs (Shang et al. 2019).

In one triplet, the relation can be regarded as a transfer function (Wang et al. 2017), and neural networks are effective tools for fitting complex functions (LeCun, Bengio, and Hinton 2015). In order to combine the two advantages, this paper proposes a new kind of knowledge graph embedding model called ParamE. In ParamE, head entity embeddings, relation embeddings and tail entity embeddings are regarded as the input, parameters, and output of a neural network. The proposed model becomes much more expressive and translational because parameters do well in converting input to output. Regarding neural network parameters as relation embeddings can help head entity embeddings and relation embeddings have more multilevel interaction. In Figure 1, we compare ParamE with ConvE (Dettmers et al. 2018). Head entities and relations are fed into neural networks together in ConvE, but ParamE considers relations as transfer function parameters rather than the input.

Apart from regarding relations as network parameters, the proposed model utilizes three different kinds of neural network architectures: multilayer perceptrons (MLP) (Goodfellow, Bengio, and Courville 2016), convolution layers (Krizhevsky, Sutskever, and Hinton 2012) and gate structure layers (Hochreiter and Schmidhuber 1997), called ParamE-MLP, ParamE-CNN, ParamE-Gate, respectively. Experiments show the proposed model can achieve good performance with different network architectures.

According to our statistics in Table 4 and Table 5, the relation frequency has an obvisious uneven distribution in the two datasets. The largest number relation may even be ten times or one hundred times than other relations. However, experiments show the problem of unbalanced distribution does not have a distinct influence on ParamE.

In summary, the proposed model regards neural network parameters as relation embeddings, such that the model has stronger expressiveness and translational properties. Since entities and relations are different kinds of words, entities and relations in ParamE are embedded into feature space and parameter space, respectively. Experiments show the proposed model is effective comparing with the existing state-of-the-art model, such as ConvE, SACN, RotatE and D4-STE/Gumbel. Our contributions are as follows:

- To the best knowledge of us, this paper is the first to regard neural network parameters as relation embeddings, which can make the proposed model much more expressive and translational.
- The proposed model considers entity embeddings and relation embeddings are from different vector spaces(feature space and parameter space), which is in line with the fact that entities and relations are different kinds of words.
- This paper implements ParamE with multilayer perceptrons, convolution layers and gate structure layers, and experiments show ParamE is a general model for different network architectures.

Related Work

In this section, we introduce several kinds of KG embedding models, such as translational models, ConvE (Dettmers et al. 2018), and other KG embedding models with deep learning methods.

Several typical translational models are TransE (Bordes et al. 2013), DistMult (Yang et al. 2014), ComplEx (Trouillon et al. 2016) and Rotate (Sun et al. 2019). TransE is an initial translational model. TransE considers the addition operation between the head entity and the tail entity. DistMul uses a bilinear diagonal model and weighted element-wise dot products to learn embeddings. ComplEx improves DistMul by complex-valued embeddings in order to better model asymmetric relations. In ComplEx, entity and relation embeddings lie in a complex space rather than a real one. RotatE could be taken as an extension of ComplEx, because it defines each relation as a rotation from the source entity to the target entity in the complex vector space so as to model and infer various relation patterns including: symmetry/antisymmetry, inversion and composition. The operations of these translational model are mainly addition and multiplication, which results in less expressive KG embeddings though they are easier to train and require fewer parameters.

Different from translational models, ConvE benefits from strong expresiveness of neural networks (Dettmers et al. 2018). In ConvE, the embeddings of head entities and relations are concatenated and reshaped into an input matrix, then the matrix is fed to multiple convolution layers. Different convolutional filters generate multiple feature map tensors to get the global information, next the tensors are projected to the embedding dimension via a fully connected layer. In the end, the tensors have a similarity matching with all entities embeddings to get the tail entity.

There are also other knowledge graph embedding models which utilize deep learning methods. NTN (Socher et al. 2013) combines head and tail entities by a relation-specific tensor and maps them to a non-linear hidden layer. NAM (Liu et al. 2016) achieves semantic matching with a deep neural network, and it concatenates the head and relation embeddings in the input layer and feed them into a deep neural network. KBGAN (Cai and Wang 2017) uses generate adversarial networks to help improve the performances of a wide range of existing knowledge graph embedding models. In order to absord graph structure information for knowledge graph embedding, SACN utilizes knowledge graph connectivity structure, node attributes and relation types by a weighted graph convolutional network (Shang et al. 2019).

Traditional translational models make the best of translational property, but are not expressive enough. On the contrary, ConvE makes full use of neural network modeling ability, but ignores the translational property. The translational property and nonlinearity fitting ability of neural networks are both essential in KG embedding. Regarding the neural network parameters as relations embedding helps us to take the two advantages both.

Method

The main idea of the proposed model is to regard neural network parameters as relation embeddings, which makes ParamE much more expresive and translational. Other than neural network parameters, the architectures also play an import role in the performances of neural networks. In order to confirm whether ParamE is a general architecture for different architectures, we implement ParamE with three different architectures: multilayer perceptrons, convolution layers, and gate structure layers, called ParamE-MLP, ParamE-CNN, ParamE-Gate. In this section, we introduce the three ParamE models.

ParamE-MLP

Multilayer perceptrons, also called feedforward neural networks, are the typical deep learning methods. A feedforward network defines a mapping $\mathbf{y} = f(\mathbf{h}; \mathbf{r})$ and learns the value of the parameters $\mathbf{r}$ that result in the best function approximation (Goodfellow, Bengio, and Courville 2016), where $\mathbf{h}$ represents the input.

In ParamE-MLP, $\mathbf{h}$ is the head entity embedding, $\mathbf{r}$ is the relation embedding, and $\mathbf{y}$ is the interaction result of head entity embedding and relation embedding. Next, $\mathbf{y}$ has a similarity matching with all entity embeddings, and we expect the matching of $\mathbf{y}$ and the tail entity embedding gets the highest score.

Specifically, ParamE-MLP uses a three layers feedfordward neural network. The whole process is shown in the followings:

$$\mathbf{y}_1 = \mathbf{f}(\mathbf{W}_0\mathbf{h} + \mathbf{b}_0) \tag{1}$$

$$\mathbf{y}_2 = \mathbf{f}(\mathbf{W}_1\mathbf{y}_1 + \mathbf{b}_1) \tag{2}$$

$$\mathbf{y} = \mathbf{f}(\mathbf{W}_2\mathbf{y}_2 + \mathbf{b}_2) \tag{3}$$

where $\mathbf{h} \in \mathbb{R}^d$ represents the input head entity embedding, $\mathbf{W}_0 \in \mathbb{R}^{d_1 \times d}$, $\mathbf{W}_1 \in \mathbb{R}^{d_2 \times d_1}$, $\mathbf{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are the weights of the first, second, third hidden layers respectively; $\mathbf{b}_0 \in \mathbb{R}^{d_1}$, $\mathbf{b}_1 \in \mathbb{R}^{d_2}$, $\mathbf{b}_2 \in \mathbb{R}^{d_3}$, are the biases of the first, second, third hidden layers respectively; d is the embedding dimension, d_1, d_2, d_3 are the dimensions of the first, second, third hidden layers, $\mathbf{f}$ represents the nonlinear function (Relu) (Glorot, Bordes, and Bengio 2011).

Taking the triplet (*Chatou, isLocatedIn, France*) for example, the embedding of the head entity (*Chatou*) is $\mathbf{h}$, the embedding of the relation (*isLocatedIn*) is a set consisting of : $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$; then we get $\mathbf{y}$, the interaction result of the head entity embedding and relation embedding, $\mathbf{y}$ is projected into the embedding dimension and has a similarity matching with the embedding of the tail entity $\mathbf{t}$ (*France*) via an inner product:

$$\psi_{\mathbf{score}} = (\mathbf{W}\mathbf{y} + \mathbf{b})\mathbf{t} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{d \times n}$ and $\mathbf{b} \in \mathbb{R}^d$ are the weight and bias of the linear projection layer respectively, and they are independent of relations and shared by all the triplets, $\psi_{\mathbf{score}}$ is the score of the triplet.

ParanE-CNN

Convolution networks are a specialized kind of neural network for processing data that has a grid-like topology (Goodfellow, Bengio, and Courville 2016), and they are tremendously successful in practical applications (Krizhevsky, Sutskever, and Hinton 2012). Using 2D convolution rather than 1D convolution can increase the expresiveness of the model through additional points of interaction between embeddings (Dettmers et al. 2018).

In ParamE-CNN, the head entity embedding $\mathbf{h}$ is first reshaped into a matrix, then the matrix has an interaction with two convolution layers, the relation embedding $\mathbf{r}$ is the set of the convolution layer weights. The specific process is as followings:

$$\mathbf{y}_1 = \mathbf{f}(\mathbf{h} * \Omega_0) \tag{5}$$

$$\mathbf{y} = \mathbf{vec}(\mathbf{f}(\mathbf{y}_1 * \Omega_1)) \tag{6}$$

where $\Omega_0 \in \mathbb{R}^{l_1 \times 1 \times n_1 \times n_2}$ represent the parameters of the first convolution layer, l_1 is the number of the output channel, k_1 and k_2 are the sizes of the convolution kernels; $\Omega_1 \in \mathbb{R}^{l_2 \times l_1 \times n_3 \times n_4}$ represent the parameters of the second convolution layer, l_1 is the number of the input channel, l_2 is the number of the output channel, k_3 and k_4 are the sizes of the convolution kernels; $\mathbf{f}$ is the activation function; the operation $\mathbf{vec}$ is to reshape a tensor into a vector, $*$ represents the conv olution operation.

In the triplet (*Chatou, isLocatedIn, France*), the embedding of the head entity (*Chatou*) is $\mathbf{h}$, the relation *isLocatedIn* embedding is the set of Ω_0 and Ω_1. The subsequent projection layer and similarity matching is similar with ParamE-MLP.

ParamE-Gate

The gate structure plays an essential role in LSTM (Hochreiter and Schmidhuber 1997) and GRU (Cho et al. 2014), and it is also widely applied in applications (LeCun, Bengio, and Hinton 2015). The core idea of the gate structure is to let information optionally (Jozefowicz, Zaremba, and Sutskever 2015). The gate structure can learn to adaptively and nonlinearly filter information, thus can help models to learn something more useful.

In ParamE-Gate, $\mathbf{h}$ still represents the head entity embedding, and one gate structure is used to filter information, the specific information flow is as followings:

$$\mathbf{y}_1 = \sigma(\mathbf{W}_0\mathbf{h}) \tag{7}$$

$$\mathbf{y}_2 = \mathbf{tanh}(\mathbf{W}_1\mathbf{h}) \tag{8}$$

$$\mathbf{y} = (1 - \mathbf{y}_1) \odot \mathbf{y}_2 \tag{9}$$

where σ is the sigmoid function, $\mathbf{h} \in \mathbb{R}^d$, $\mathbf{W}_0 \in \mathbb{R}^{d_1 \times d}$, $\mathbf{W}_1 \in \mathbb{R}^{d_1 \times d}$ represent the correspongding weights, $\odot$ is a Hadamard product.

For the triplet (*Chatou, isLocatedIn, France*), the entity (*Chatou*) embedding is $\mathbf{h}$, and the relation (*isLocatedIn*) embedding is the set of $\mathbf{W}_0, \mathbf{W}_1$. In Equation 7, the sigmoid operation sets every element in $\mathbf{y}_1$ between 0 and 1. Then in Equation 9, $(1 - \mathbf{y}_1)$ is the gate and controls how much information of $\mathbf{y}_2$ is allowed to pass. The subsequent projection layer and similarity matching are also similar with ParamE-MLP.

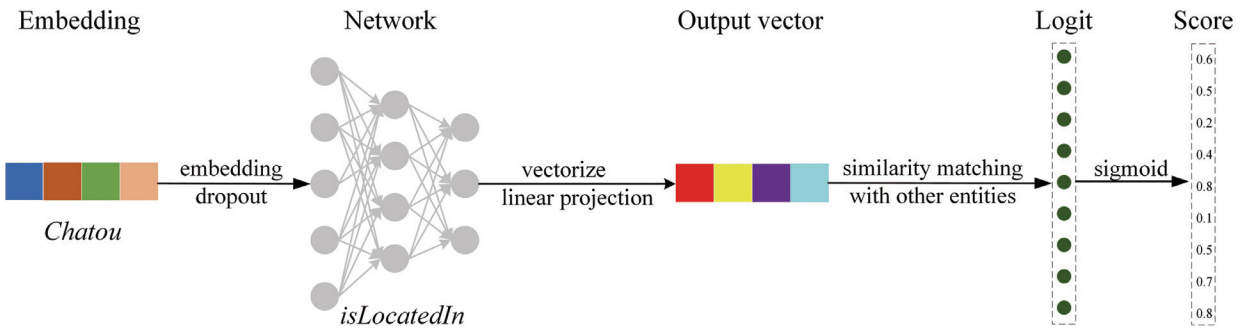

Figure 2: An illustration of the proposed ParamE. Taking the triplet (*Chatou, isLocatedIn, France*) for example, in ParamE, the network first loads the relation embeddings (*isLocatedIn*) as its parameters, then the head entity embedding (*Chatou*) is fed into the network and the output is vectorized and projected to the embedding dimension, through a linear layer. Then the result has a matrix multiplication with all the entity embeddings to get the logits. Finally, the score is generated after a sigmoid function on the logit.

Training

Although ParamE-MLP, ParamE-CNN, ParamE-Gate have different network architectures, the main ideas of them are the same, which are to regard neural network parameters as relation embeddings. They can be summrized in the following scoring function:

$$\psi_{\mathbf{score}} = ((\mathbf{f_{nn}}(\mathbf{h};\mathbf{r}))\mathbf{W} + \mathbf{b})\mathbf{t} \tag{10}$$

where $\mathbf{f_{nn}}$ represents a neural network, ParamE-MLP corresponds to multilayer perceptrons, ParamE-CNN corresponds to convolution layers , ParamE-Gate corresponds to and gate structure layers, more network architectures can be explored in the future, $\mathbf{h}$ is the head entity embedding and the input to a neural network, $\mathbf{r}$ is both the relation embedding and neural network parameters, $\mathbf{W}$ is the parameters of the projection layer, $\mathbf{t}$ is the tail entity embedding.

After we get the scores of the true triplets and false triplets, we use a sigmoid function $\sigma(\cdot)$ to set every score between 0 and 1, which can be represented as

$$p = \sigma(\psi_{\mathbf{score}}) \tag{11}$$

the loss function is the binary cross entropy loss, which is the same with ConvE (Dettmers et al. 2018). The whole information flow process is shown in Figure 2

Since the ParamE is relation-based, the training process is different from previous models, and the ParamE is trained relation by relation. We first divide the triplets into different groups according to the relation type, in other words, the triplets in one group has the same relation. We then calculate the ratio of the number of triples in each group to the number of all triples, and the training process is group by group. For one epoch, the number of iteration is $\mathbf{n}$, and the number of batch size is $\mathbf{b}$. When training in one epoch, we randomly pick up one group to train based on the ratio of each group, and the number of selections in one epoch is $\mathbf{n}$. For one iteration, we randomly select $\mathbf{b}$ triplets from the corresponding group as the input, and the network loads the relation embeddings as its parameters. After one iteration training, the network parameters are saved for next loading.

In a word, we implement ParamE with three different architectures: multilayer perceptrons, convolution layers and gate structure layers. ParamE-MLP, ParamE-CNN, and ParamE-Gate show the ParamE is a general model for different networks, and more reasonable network architectures can be designed in the future.

Experiments

Background

Formally, if we take $\mathcal{E}$ and $\mathcal{R}$ as the sets of entities and relations, respectively. A knowledge graph can be represented as $\mathcal{G} = \{(\mathbf{h},\mathbf{r},\mathbf{t})\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For each $(\mathbf{h},\mathbf{r},\mathbf{t}) \in \mathcal{G}$, we denote the reverse of $\mathbf{r}$ by $\mathbf{r}^{-1}$, and add an additional triplet $(\mathbf{t},\mathbf{r}^{-1},\mathbf{h})$ to $\mathcal{G}$. For example, for the triplet (*Chatou, isLocatedIn, France*), we add an additional triplet (*France, isLocatedIn_reverse, Chatou*) to $\mathcal{G}$. In addition, we replace the head or tail entity with other entities to generate false triplets for evaluate the embedding model.

There are many models for knowledge graph embedding, such as TransE (Bordes et al. 2013), ConvE (Dettmers et al. 2018), and we summarize the scoring functions and embedding parameter shapes of several representative knowledge graph embedding models in Table 1 to compare with ParamE.

Evaluation Protocol

The target of knowledge graph embedding is to learn a good representation of entities, relations, and a scoring function, such that we can predict a triple $(\mathbf{h},\mathbf{r},\mathbf{t})$ with $\mathbf{h}$ or $\mathbf{t}$ unknown. When given $(\mathbf{h},\mathbf{r})$, we should predict the correspongding $\mathbf{t}$; or given $(\mathbf{r},\mathbf{t})$, we are to get the correspongding $\mathbf{h}$, which becomes predicting $\mathbf{h}$ when given $(\mathbf{t},\mathbf{r}^{-1})$. In the experiments, this papers take one $(\mathbf{h},\mathbf{r})$ pair and score it against all entities simultaneously, same with ConvE (Dettmers et al. 2018).

In our experiments, we use the proportion of correct triples ranked in top 1,3,10, and mean reciprocal rank, the corresponding symbols in the results tables are Hits@1,

Table 1: Scoring Functions $\psi_r(\mathbf{h}, \mathbf{t})$ of different knowledge graph embedding model for link prediction, where $\odot$ denotes the Hadamard product, $\bar{\cdot}$ denotes conjugate for complex vectors, $*$ denotes 2D convolution, $\mathbf{f}$ denotes activation function, $\mathbf{f_{mlp}}(\mathbf{h}; \mathbf{r})$ denotes three layers perceptrons, $\mathbf{h}$ is the input of three layers perceptrons, $\mathbf{r}$ is the parameters of three layers perceptrons, composed of $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$, σ denotes a sigmoid function.

Model	Scoring Function $\psi_r(\mathbf{h}, \mathbf{t})$	Embedding Parameters
TransE(Bordes et al. 2013)	$-\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
DistMult(Yang et al. 2014)	$\langle \mathbf{r}, \mathbf{h}, \mathbf{t} \rangle$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
ComplEx(Trouillon et al. 2016)	$\mathrm{Re}(\langle \mathbf{r}, \mathbf{h}, \bar{\mathbf{t}} \rangle)$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$
ConvE(Dettmers et al. 2018)	$\mathbf{f}(\mathrm{vec}(f([\bar{\mathbf{r}}, \bar{\mathbf{h}}] * \mathbf{\Omega}))\mathbf{W})\mathbf{t}$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
SACN(Shang et al. 2019)	$\mathbf{f}(\mathrm{vec}\left(\mathbf{M}\left(\mathbf{h}, \mathbf{r}\right)\right)\mathbf{W})\mathbf{t}$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
RotatE(Sun et al. 2019)	$-\|\mathbf{h} \odot \mathbf{r} - \mathbf{t}\|^2$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k, \|r_i\| = 1$
ParamE-MLP	$(\mathbf{W}\mathbf{f_{mlp}}(\mathbf{h}; \mathbf{r}) + \mathbf{b})\mathbf{t}$	$\mathbf{h}, \mathbf{t} \in \mathbf{R}^{k_0}, \mathbf{r} = (\mathbf{W_i} \in \mathbb{R}^{k_{i+1} \times k_i}, \mathbf{b_i} \in \mathbb{R}^{k_{i+1}}), i = 0, 1, 2$
ParamE-CNN	$\mathbf{W}(\mathbf{vec}(\mathbf{f}(\mathbf{f}(\mathbf{h} * \Omega_0) * \Omega_1)) + \mathbf{b})\mathbf{t}$	$\mathbf{h}, \mathbf{t} \in \mathbf{R}^k, \mathbf{r} = (\Omega_0 \in \mathbb{R}^{l_1 \times 1 \times n_1 \times n_2}, \Omega_1 \in \mathbb{R}^{l_2 \times l_1 \times n_1 \times n_2})$
ParamE-Gate	$(\mathbf{W}((1 - \sigma(\mathbf{W}_0\mathbf{h})) \odot \mathbf{tanh}(\mathbf{W}_1\mathbf{h})) + \mathbf{b})\mathbf{t}$	$\mathbf{h}, \mathbf{t} \in \mathbf{R}^k, \mathbf{r} = (\mathbf{W}_0 \in \mathbb{R}^{k_1 \times k}, \mathbf{W}_1 \in \mathbb{R}^{k_1 \times k})$

Table 2: Statistics of different datasets

Dataset	FB15k-237	WN18RR
Entities	14541	40943
Relations	237	11
Training	272115	86835
Validation	17535	3034
Test	20466	3134
All	310116	93003

Hits@3, Hits@10, MRR. Similar to previous work (Bordes et al. 2013), we use a filtered setting,i.e we remove corrupt triples which are already present in the or validation or test sets.

Datasets

We use two benchmark datasets(FB15k-237 and WN18RR) to evaluate the performances of ParamE-MLP, ParamE-CNN, ParamE-Gate.

FB15k-237 FB15k-237 (Toutanova and Chen 2015) dataset contains 310116 triples with 14541 entities and 237 types of relations. As a subset of FB15K (Bordes et al. 2013), originally derived from Freebase, FB15k-237 does not consist of inverse relations.

WN18RR WN18RR (Dettmers et al. 2018) contains 93003 triples with 40943 entities and 11 kinds of relations. WN18RR is a subset of WN18 (Bordes et al. 2013) which is from WordNet. Comparing with WN18, WN18RR dose not suffer inverse relation test leakage.

The details of the two datasets are listed in Table 2.

Experimental Setup

Sine ParamE is relation-based, relation embeddings are parameters of neural networks. For one time training, we load the relation embeddings as the network parameters, so we have to train triplets with the same relation one time. We first divide all the triplets into different groups based on the type of relations in triplets. In other words, triplets in one group have the same relation. As is shown in Table 4 and Table 5, the numbers of different relation triplets vary greatly, if the number of one group of triplets is large, we need to give them more chances to train. So we calculate the ratio of the number of different groups of triplets to the total number of triplets.

There are 237 types of relations in FB15k-237 dataset, and 11 types of relation in WN18RR dataset. As a result, the number of iteration in FB15k-237 dataset should be larger than in WN18RR dataset, and the batch size in FB15k-237 dataset should be smaller than in WN18RR dataset. For each epoch, we randomly select one relation to train according to its ration, and the number of selections is the iteration; once we get one relation, we randomly choose some triplets within the group with the selected relation, the number of triplets is the batch size.

The learning rate decay is utilized for quicker convergence, after several epochs, the current learning rate multiplies a decay factor. For WN18RR, after 3 epochs, the current learning rate multiplies 0.99; for FB15k-237, the decay period is 10 epochs, and the decay factor is 0.8. The label smoothing (Szegedy et al. 2016) is also used to reduce overfitting.

For WN18RR dataset and FB15k-237 dataset, the embedding dimension of entities is different in different neural network architectures. In ParamE-MLP, the dimension of the first, second, third hidden layers are 200, 400, 800, respectively. After each hidden layer, there are dropout (Srivastava et al. 2014) operations to release overfitting. In ParamE-CNN, the number of output channel in the first convolution layer is 32, and the number of output channel in the second convolution layer is 64; the kernel size in the two convolution layers is 3×3.

We use the adaptive moment(Adam) algorithm (Kingma and Ba 2014) for training these models. In ParamE-CNN, we use batch normalization (Ioffe and Szegedy 2015) and layer normalization (Ba, Kiros, and Hinton 2016) in ParamE-Gate to reduce training time and increase rate of convergence.

Result

The results of the ParamE models on the standard FB15k-237 dataset, WN18RR dataset are shown in Table 3. We compare ParamE-MLP, ParamE-CNN, ParamE-Gate with

Table 3: Results of link prediction for FB15k-237, WN18RR

Model	FB15k-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE(Bordes et al. 2013)	0.294	-	-	0.465	0.226	-	-	0.501
DistMult(Yang et al. 2014)	0.241	0.155	0.263	0.419	0.43	0.39	0.44	0.49
ComplEx(Trouillon et al. 2016)	0.247	0.158	0.275	0.428	0.44	0.41	0.46	0.51
ConvE(Dettmers et al. 2018)	0.325	0.237	0.356	0.501	0.43	0.40	0.44	0.52
SACN(Shang et al. 2019)	0.36	0.27	0.40	0.55	0.47	0.43	0.48	0.54
RotatE(Sun et al. 2019)	0.338	0.241	0.375	0.533	0.476	0.428	0.492	**0.571**
D4-STE(Xu and Li 2019)	0.320	0.230	0.353	0.502	0.480	0.452	0.491	0.536
D4-Gumbel(Xu and Li 2019)	0.300	0.204	0.332	0.496	0.486	0.442	0.505	0.557
ParamE-MLP	0.314	0.240	0.339	0.459	0.407	0.384	0.429	0.445
ParamE-CNN	0.393	0.304	0.426	**0.576**	0.461	0.434	0.472	0.513
ParamE-Gate	**0.399**	**0.310**	**0.438**	0.573	**0.489**	**0.462**	**0.506**	0.538

several state-of-the-art models, including TransE (Bordes et al. 2013), DistMult (Yang et al. 2014) , ComplEx (Trouillon et al. 2016), ConvE (Dettmers et al. 2018) , SACN (Shang et al. 2019), RotatE (Sun et al. 2019), D4-STE(Xu and Li 2019), and D4-Gumbel(Xu and Li 2019).

We can see that ParamE-Gate nearly outperfoms all the other models. In FB15k-237 dataset, the ParamE-Gate model is 3.9% higher than SACN's MRR, 4% higher than SACN's Hits@1, 3.7% higher than SACN's Hits@3, in ParamE-CNN, the Hits@10 is 2.6% higher than SACN's Hits@10. In WN18RR dataset, ParamE-Gate improves upon D4-Gumbel's MRR by a margin of 0.3%, upon D4-STE's Hits@1 by a margin of 1%, upon D4-Gumbel's Hits@3 by a margin of 0.1%.

Due to the differences of the FB15k-237 dataset and WN18RR dataset, some models may get high performances in FB15k-237, like SACN, but not suitable for the WN18RR dataset. While other models are the opposite, such as RotatE, D4-STE and D4-Gumbel. The ParamE-CNN prefers FB15k-237 dataset rather than WN18RR dataset. However, ParamE-Gate can work well in the both datasets.

Comparing with other evaluation criterion, the ParamE-Gate works better on Hits@1 than Hits@3,10 or MRR. In FB15k-237 compared with SACN, the ParanE-Gate improves 4.0% on Hits@1, much better than 2.6% on Hits@10, so the same as in WN18RR. In the training process, head entity embeddings are like input features, and tail entity embeddings are like output, so the network tends to learn the mapping between the head entity and the most likely correct tail entity. As a result, the improvement on Hits@1 of the ParamE is more obvisious than others.

ConvE and ParamE-CNN have similar network architectures, and the difference is that the relation embeddings are the input in ConvE, but the parameters in ParamE-CNN. Experiments on the both models show the ParamE-CNN can capture the intrinsic property and is more reasonable.

For the ParamE-MLP, ParamE-CNN, ParamE-Gate, the ParamE-MLP gets the worst performance because MLP has a weaker modeling ability than convolution layers and gate structure. Although convolution layers are good at extracting feature, the head entity embeddings do not have spatial structure, therefore ParamE-CNN model dose not get the best performance. Since the gate structure can optionally let information through, useful information is allowed to pass while the rest is filtered, so the ParamE-Gate is more effective.

Analysis of performances of different kinds of relations

In order to make better use of the nonlinerailty ability of neural networks, we regard networks parameters as relation embeddings, which is the advantage of our model. However, this method may have some disadvantages. We all know training a good neural network needs lots of data. In the two datasets, the number of triplets belonging to different relations varies a lot. As is shown in Table 4, there are totally 11 relations in WN18RR, and we list the number of triplets belonging to different relations. We can see relation '_hypernym' has 34796 triplets in the training set, 1251 triplets in the test set; relation '_similar_to' only has 80 triplets in the training set, 3 triplets in the test set. Table 5 shows information in FB15k-237, since there are 237 kinds of relations, so we randomly select 11 relations. The relation with the most triplets can have 10945 triplets in the training set, 1311 triplets in the test set; the relation with fewer triplets only has 264 triplets in the training set, 18 triplets in the test set.

We list the ParamE-Gate performance of different relations in Table 4 and Table 5 to check whether the relation with more triplets has a better result than the relation with fewer triplets. The model is trained with all the triplets and tested on every single relation repsectively. In Table 4, relation '_hypernym' with the most triplets gets a poor performance (MRR 0.196, Hits@1 0.161, Hits@3 0.214, Hits@10 0.261), relation '_derivationally_related_form' with the second most triplets has a much better result (MRR 0.957, Hits@1 0.953, Hits@3 0.961, Hits@10 0.964). Relation '_verb_group' with only 1138 triplets in the training set has the best result (MRR 0.975, Hits@1 0.974, Hits@3 0.974, Hits@10 0.974), however relation '_member_of_domain_region' with nearly the same number as '_verb_group' has a much worse result (MRR 0.170,

Table 4: The numbers and performances of different kinds of relation in WN18RR.

Relation Name	training number	test number	MRR	Hits@1	Hits@3	Hits@10
_hypernym	34796	1251	0.196	0.161	0.214	0.261
_derivationally_related_form	29715	1074	0.957	0.953	0.961	0.964
_member_meronym	7402	253	0.139	0.080	0.158	0.241
_has_part	4816	172	0.124	0.076	0.156	0.203
_synset_domain_topic_of	3116	114	0.553	0.491	0.596	0.667
_instance_hypernym	2921	122	0.594	0.508	0.648	0.754
_also_see	1299	56	0.598	0.554	0.643	0.643
_verb_group	1138	39	0.975	0.974	0.974	0.974
_member_of_domain_region	923	26	0.170	0.12	0.27	0.27
_member_of_domain_usage	629	24	0.025	0	0	0.125
_similar_to	80	3	0.186	0	0.333	0.333

Table 5: The numbers and performances of different kinds of relation in FB15k-237.

Relation Name	training number	test number	MRR	Hits@1	Hits@3	Hits@10
/people/person/profession	10945	1311	0.651	0.519	0.738	0.891
/music/genre/artists	5880	664	0.164	0.107	0.179	0.274
/film/film/language	2570	314	0.763	0.669	0.780	0.904
/music/record_label/artist	2226	266	0.062	0.023	0.052	0.127
/film/film/country	2407	131	0.690	0.580	0.756	0.893
/film/film/written_by	787	26	0.029	0	0.038	0.077
/people/person/languages	783	98	0.765	0.694	0.796	0.929
/film/film_subject/films	603	77	0.005	0	0	0
/sports/sports_team/sport	423	53	0.972	0.962	0.981	0.981
/film/film/story_by	394	31	0.253	0.161	0.323	0.387
/film/film/film_festivals	264	18	0.542	0.333	0.722	0.889

Hits@1 0.12, Hits@3 0.27, Hits@10 0.27). In FB15k-237 dataset from Table 5, we can also see the relation with many triplets can have a bad result, the relation with few triplets may have a good performance.

From the analysis above, there is no obvisious correlation between the number of triplets and the performance for every relation in the ParamE-Gate. This is mainly because the ParamE-Gate can extract the features of the entire knowledge graph and learn effective entity embeddings.

Conclusion and Future Work

We developed a new kind of knowledge graph embedding method for link prediction called ParamE. The key idea of ParamE is to regard neural network parameters as relation embeddings. Compared with some neural network based models, ParamE makes use of the translational properties of relations. ParamE has a stronger expresiveness than traditional translational models. In addition, ParamE regards entitiy embeddings and relation embeddings are from different vector space, i.e., one is feature space and the other is parameter space, which is very reasonable, since entities and relations are totally different kinds of words. We implement ParamE with three different neural network architectures: multilayer perceptrons(MLP) (Goodfellow, Bengio, and Courville 2016), convolution layers (Krizhevsky, Sutskever, and Hinton 2012) and gate structure layers (Hochreiter and Schmidhuber 1997), called ParamE-MLP, ParamE-CNN, ParamE-Gate, respectively. The experment results on the three ParamE models show the ParamE is a general method for different network architectures, more reasonable architectures can be explored in the future. The ParamE models have an obvisious improvement than the previous state-of-the-art meths such as SACN (Shang et al. 2019) , RotatE (Sun et al. 2019), D4-STE(Xu and Li 2019), and D4-Gumbel(Xu and Li 2019) .

In the future work, we would like to incorporate graph structure information into the ParamE models, such as the neighbor entities. Combining graph neural network with the ParamE models may be a good choice. In addition, we would also like to make a study of some relations with poor performance in Table 4 and Table 5. Some well-designed neural networks may be able to deal with such relations.

Acknowledgments

This work is supported by the National Key Research and Development Plan of China (No.2018YFB1005003), and National Natural Science Foundation of China (No.61425017, No.61831022, No.61771472, No.61773379).

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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.

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.

Cai, L., and Wang, W. Y. 2017. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071*.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; and McCallum, A. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 315–323.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

He, H.; Balakrishnan, A.; Eric, M.; and Liang, P. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. *arXiv preprint arXiv:1704.07130*.

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

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jozefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, 2342–2350.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Liu, Q.; Jiang, H.; Evdokimov, A.; Ling, Z.-H.; Zhu, X.; Wei, S.; and Hu, Y. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. M. 2013. Yago3: A knowledge base from multilingual wikipedias.

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

Nguyen, D. Q. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*.

Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; and Zhou, B. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 3060–3067.

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.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Toutanova, K., and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 57–66.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, 2071–2080.

Wang, Q.; Mao, Z.; Wang, B.; and Guo, L. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29(12):2724–2743.

Xiong, C.; Power, R.; and Callan, J. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, 1271–1279.

Xu, C., and Li, R. 2019. Relation embedding with dihedral group in knowledge graph. *arXiv preprint arXiv:1906.00687*.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.