ProjR: Embedding Structure Diversity for Knowledge Graph Completion

Wen Zhang, Juan Li, and Huajun Chen(✉)

Zhejiang University, Hangzhou, China
{wenzhang2015,lijuan18,huajunsir}@zju.edu.cn

Abstract. Knowledge graph completion aims to find new true links between entities. In this paper, we consider an approach to embed a knowledge graph into a continuous vector space. Embedding methods, such as TransE, TransR and ProjE, are proposed in recent years and have achieved promising predictive performance. We discuss that a lot of substructures related with different relation properties in knowledge graph should be considered during embedding. We list 8 kinds of substructures and find that none of the existing embedding methods could encode all the substructures at the same time. Considering the structure diversity, we propose that a knowledge graph embedding method should have diverse representations for entities in different relation contexts and different entity positions. And we propose a new embedding method ProjR which combines TransR and ProjE together to achieve diverse representations by defining a unique combination operator for each relation. In ProjR, the input head entity-relation pairs with different relations will go through a different combination process. We conduct experiments with link prediction task on benchmark datasets for knowledge graph completion and the experiment results show that, with diverse representations, ProjR performs better compared with TransR and ProjE. We also analyze the performance of ProjR in the 8 different substructures listed in this paper and the results show that ProjR achieves better performance in most of the substructures.

Keywords: Diversity structures · Knowledge graph embedding
Knowledge graph completion

1 Introduction

Knowledge graphs (KGs) are built to represent knowledge and facts in the world and have become useful resources for many artificial intelligence tasks such as web search and question answering. A knowledge graph could be regarded as a multi-relational directed graph with entities as nodes and relations as labeled edges. An instance of an edge is a fact triple in the form of (h, r, t) and h, r, t denote the head entity, relation, and tail entity respectively. For example, (Steve Jobs, $isFounderOf$, Apple Inc.) represents the fact that Steve Jobs is the

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11108, pp. 145–157, 2018.
https://doi.org/10.1007/978-3-319-99495-6_13

founder of company Apple. Many huge knowledge graphs have been built automatically or semi-automatically in recent years, such as Yago [19], WordNet [13] and Google Knowledge Graph[1]. Though typical knowledge graphs may contain more than millions of entities and billions of facts, they still suffer from incompleteness.

Much work has focused on knowledge graph completion. Some tried to extract more triples through large text corpora, such as OpenIE [1] and DeepDive [26]. Others tried to get new triples by reasoning based on the existing knowledge graph. Traditional reasoning methods including ontology inference machine, such as Pellet[2], which relies on a well-defined ontology. Knowledge graph embedding is another way to complete knowledge graphs. It tries to embed all the elements in a knowledge graph, including entities and relations, into a continuous vector space while preserving certain properties of the original KG. KG embedding makes it possible to complete the knowledge graph through implicit inference by calculations between the vector representations.

We regard the basic process of most knowledge graph embedding methods as two steps: (1) get combined representation of the given head entity-relation pair through a combination operator $C(h, r)$ and then (2) compare the similarity between candidate tail entities and the combined representation through a similarity operator $S(C(h, r), t)$. The goal is to make the similarity between true candidate entities and combined representation as large as possible and the similarity for false candidate entities as small as possible.

There are a lot of knowledge graph embedding methods proposed in the last few years. One of the simple but effective method is TransE [4] which represents each entity and relation as a vector. The combination operator is defined as an addition and the similarity operator is defined based on distance. ProjE [16] is another knowledge graph embedding method whose basic idea is almost the same with TransE. ProjE also assigns each entity and relation with one vector. The combination operator of ProjE is a global linear layer and the similarity operator is defined as a dot product between the candidate vector and combined representation. ProjE performs better than TransE in link prediction task because of the different definition of the final loss function.

Considering the different properties of relations, improved methods, such as TransH [21], TransR [12] and TransD [9], are proposed based on TransE. Those methods inspired us to consider the structure diversity of knowledge graph as relations of different properties always correspond to different graph structures. We list 8 kinds of graph structures in this paper and find that none of the previous embedding methods can properly encode all of these structures at the same time. Some methods such as TransR can encode N-1, N-N, 1-N-1 structures but not one-relation-circle structures. Some methods, such as ProjE, could encode one-relation-circle structures but not N-1, N-N, 1-N-1 structures.

[1] https://www.google.com/intl/en-419/insidesearch/features/search/knowledge.html.
[2] http://pellet.owldl.com/.

The reason why TransR can encode 1-N, N-N, and 1-N-1 structures is that before calculating the similarity, TransR uses a relation specific matrix to project entity vector to relation space and get a new entity representation. In other words, every entity in TransR has one vector representation under each different relation context. The reason why ProjE can encode one-relation-circle structures is that the vector representation for one entity under different entity position (as a head entity or as a tail entity) is different. So the key point to enable embedding methods to encode the diversity of knowledge graph is that every entity should have diverse representations in different relation contexts and entity positions.

In this paper, we propose a new embedding method ProjR which combines TransR and ProjE together. To achieve diverse representation for entities in different relation contexts, we define a unique combination operator for each relation in ProjR. To achieve diverse representations for entities under different entity position, we follow the process of ProjE to project head entity vector through a matrix during combination process and not project tail entity during similarity operator. We conduct link prediction experiments on knowledge graph completion benchmark datasets and evaluate the results in the same way as previous work. The evaluation results show that ProjR achieves better predictive performance compared with both TransR and ProjE, and also some other embedding methods. We also analyze the performance of ProjR with the 8 different substructures listed in Table 6 and the results show that ProjR achieves better performance in most of the substructures compared with ProjE.

The contributions of this paper are as follows:

- We list 8 kinds of graph structures to analyze the structure diversity of knowledge graph and also find examples from real knowledge graph Freebase for each structure.
- We analyze the ability of most related embedding models to encode the structure diversity and find that diverse representations for entities in different relation contexts and entity positions are helpful for encoding the structure diversity in knowledge graph.
- We propose a new knowledge graph embedding method ProjR based on the idea of diverse representations for entities. The experiment results on link prediction tasks show that ProjR achieves better performance than TransR and ProjE. The experiment results of the performance of ProjR on different structures also prove that with diverse representations for entities ProjR could handle the structure diversity of knowledge graph more properly.

2 Related Work

We summarize the related methods in Table 1 with information of the score function and the number of parameters to learn during training. The work most related to ours are TransR [12] and ProjE [16].

TransR: TransR [12] is an extended method of TransE [4]. We call them translation-based methods because their basic assumption is that the relation r

Table 1. A summary for the most related methods. The bold lower letter denotes vector representation and the bold upper letter denotes the matric representation. $\mathbf{I}$ denotes identity matrix. n_e and n_r are the number of entities and relations in knowledge base, respectively. d denotes the embedding dimension.

Model	Sore function $S(h,r,t)$		#Parameters
	$C(h,r)$	$S(h,r,t)$	
TransE	$\mathbf{h}+\mathbf{r}$	$\|C(h,r)-\mathbf{t}\|_{l_{1/2}}$	$(n_e+n_r)d$
TransH	$(\mathbf{h}-\mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r+\mathbf{r})$	$\|C(h,r)-(\mathbf{t}-\mathbf{w}_r^\top t\mathbf{w}_r)\|_2^2$	$(n_e+2n_r)d$
TransR	$\mathbf{M}_r\mathbf{h}+\mathbf{r}$	$\|C(h,r)-\mathbf{M}_r\mathbf{t}\|_2^2$	$(n_e+n_r)d+n_rd^2$
TransD	$(\mathbf{r}_p\mathbf{h}_p^\top+\mathbf{I}^{k\times k})\mathbf{h}+\mathbf{r}$	$\|C(h,r)-(\mathbf{r}_p\mathbf{t}_p^\top+\mathbf{I}^{d\times d})\mathbf{t}\|_2^2$	$2(n_e+n_r)d$
ProjE	$tanh(\mathbf{D}_1\mathbf{h}+\mathbf{D}_2\mathbf{r}+\mathbf{b}_1)$	$\sigma(C(h,r)\cdot\mathbf{t}+\mathbf{b}_2)$	$(n_e+n_r+5)d$

in triple (h,r,t) could be regarded as a translation from the head entity h to the tail entity t. TransE represents each entity and relation with one vector and make the constraint for a true triple (h,r,t) as $\mathbf{h}+\mathbf{r}\approx\mathbf{t}$. Unstructured [3] is a special case of TransE which sets all relation vectors as zero vectors. Considering the different properties of relations, TransH [21] defines a hyperplane for each relation and projects head and tail entities onto the current relation hyperplane before the calculation of distance. Different from TransE and TransH which represent all elements in the same vector space, TransR represents entities in entity space and represents relations in relation space. And entities h and t should be projected into the current relation space through the relation projection matrix $\mathbf{M}_r$. TransD [9] is an extended method of TransR which defines dynamic projection matrices related with both relations and entities. TransH, TransR and TransD all achieve diverse representations for entities in different relation contexts by different projection strategies but unable to encode one-relation-circle because there is only one representation for one entity with same relation context.

ProjE: ProjE [16] is another knowledge graph embedding method which gets the combined representation of an input head entity-relation pair through a global linear layer. Then projecting all the candidate entity vectors onto the combined vector result which could be regarded as a similarity computation. But different from translation-based methods which optimize a margin-based pairwise ranking loss, ProjE optimizes a cross entropy based ranking loss of a list of candidate entities collectively which makes ProjE more flexible with negative candidate sampling and enhance the ability to handle very large datasets. ProjE also points out that the number of negative samples will affect the embedding results obviously. ProjE is able to encode the one-relation-circle structures.

Other Methods: RESCAL [15] regards the whole knowledge graph as a multi-hot tensor and embeds the knowledge graph based on tensor factorization. NTN [17] is a neural tensor network which represents each relation as a tensor. HOLE [14] employs correlation between different dimensions of entity vectors during training of the vector representations. Some methods also combine other information together with fact triples. RTransE [6] and PTransE [11] employ the path information of 2–3 length over knowledge graph. Jointly [20], DKRL [24], TEKE [22] and SSP [23] combine unstructured entity description texts together with the structured triples during training. The external text information makes those methods more likely to cover the out-of-knowledge-graph entities. TKRL [25] considers the information of entity class. But in this paper, we only focus on the methods that only use triples' information.

3 Our Method

In this section, we first introduce 8 kinds of structures to prove the structure diversity of knowledge graph and analyze the ability of the most related embedding methods to encode these 8 kinds of structures. Then we introduce the new method ProjR.

3.1 Structure Diversity of Knowledge Graph

The complex connections between entity nodes cause the structure diversity of knowledge graph. Relations with different properties are always related to different graph structures. We introduce 8 kinds of substructures in this section.

1-1 relation structure means that one entity links to at most one entity through this relation. **1-N** relation structure means that one entity links to more than one entities through this relation. **N-1** relation structure means that there are more than one entities linking to the same entity through this relation. **N-N** relation structure means that one entity link to more than one entities through this relation and one entity also could be linked to more than one entities through this relation. Those four kinds relation properties are first proposed in [21]. **1-N-1** structure means that there are more than one relation that link one same entity to another same entity. **C1, C2, C3** are special case of one-relation-circle (ORC) substructures which is first proposed in [27]. **C1** means that one

Table 2. This table lists the ability of five most related KG embedding methods to encode the substructures and the number of vector representations for every entity and relation. n_r denotes the number of relations in knowledge graph. In the column of "types of structure", "$\surd$" means model in current row can encode the substructure in current column and "$\times$" means can't.

Method	# representations		Types of substructure							
	Entity	Relation	1-1	1-N	N-1	N-N	1-N-1	C1	C2	C3
TransE [4]	1	1	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
TransH [21]	n_r	1	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$
TransR [12]	n_r	1	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$
TransD [9]	n_r	1	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$
ProjE [16]	2	2	$\surd$	$\times$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$

entity connects to itself by one relation. **C2** means that two entity connects to each other by the same relation. **C3** means that three entities connect to each other through the same relation and the connections form a circle if ignoring the direction of relations.

In Table 2, we analyze the ability of related embedding models to encode these 8 kinds of structures. We regard one method could encode one kind structure if and only if it is possible to let the similarity score of all the true triple participating in the current structure to be nearly the maximum value. We calculate the vector representation of entities and relations following this rule: we regard any projection operation result of one entity as a representation for it. For example, with triple (h, r, t) in TransH, the head entity h and the tail entity t will be projected onto the relation r specific hyperplane before calculating the distance, which means each entity has a vector representation on every relation hyperplane. So there are n_r representations for each entity in TransH.

Although none of them could encode all the structures, we also could conclude that diverse representations for entities will improve the capability of encoding structure diversity. TransH, TransR and TransD are able to encode 1-N, N-1 and N-N relation structures because they separate the representations for entities in different relation contexts and have n_r kinds of representations for every entity. ProjE is able to encode C1, C2 and C3 because the different representations for one entity in different entity positions enable ProjE to decompose the one-relation-circle structures.

To make embedding model more powerful to encode the structure diversity, we combine TransR and ProjE and propose ProjR based on the key idea of diverse representations for entities in different relation contexts and entity positions.

3.2 ProjR

In ProjR, we define a score function to calculate the probability of an input triple (h, r, t) to be true. And we regard the probability score function as two parts: a combination operator and a similarity operator.

Combination Operator: The input of combination operator is head entity-relation pair (h, r). The head entity embedding is set as $\mathbf{h} \in \mathbb{R}^d$ and the relation embedding is set as $\mathbf{r} \in \mathbb{R}^d$. d is the dimension of embedding vectors.

To achieve diverse representations for entities in different relation contexts, ProjR defines a combination operator $C_r(h)$ for each relation r:

$$C_r(h) = \mathbf{c}_{hr} = tanh(\mathbf{D}_r^e \mathbf{h} + \mathbf{D}_r^r \mathbf{r} + \mathbf{b}_c)$$

$\mathbf{D}_r^e \in \mathbb{R}^{d \times d}$ is a diagonal matrix defined for linear transformation of head entity related with relation r. $\mathbf{D}_r^r \in \mathbb{R}^{d \times d}$ is a diagonal matrix defined for linear transformation of relation r. We choose the diagonal matrix instead of normal matrix in consideration of the balance between diverse representation ability and the number of parameters. $\mathbf{b}_c \in \mathbb{R}^d$ is a global bias vector. $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ is a nonlinear activate function in which the output value will be constrained to $(-1, 1)$. Each combination operator will project entity with a different diagonal matric and generate a specific representation for each entity.

Similarity Operator: After getting the vector result $\mathbf{c}_{hr}$ from combination operator for (h, r), ProjR calculates the similarity between the $\mathbf{c}_{hr}$ and the tail entity vector $\mathbf{t}$ as the final probability score for triple (h, r, t) to be true. To achieve the diverse representations for entities in different entity positions, we use the tail entity vector directly without any projection. Although defining another projection matrix for the tail entity related with relation r will further improve the ability of ProjR to encode diversity of structures in knowledge graph, it will also increase a lot of parameters.

Considering the convenience for computation, we define the similarity operator as follows:

$$S(h, r, t) = \sigma(\mathbf{t} \cdot \mathbf{c}_{hr})$$

$\mathbf{t} \in \mathbb{R}^d$. We use dot product to simulate the similarity between $\mathbf{c}_{hr}$ and $\mathbf{t}$. And $\sigma(z) = \frac{1}{1+e^{-z}}$ is used to constrain the final output to $(0, 1)$ as a probability score.

Loss Function: During training, we define the following learning objective:

$$\begin{aligned} L = -&\sum_{(h,r,t)\in\triangle} log(S(h, r, t)) \\ &- \sum_{(h',r,t')\in\triangle'} log(1 - S(h', r, t')) + \lambda \sum_{\mathbf{p}\in P} \|\mathbf{p}\| \end{aligned}$$

$\triangle$ is the set of positive triples in training data. And $\triangle'$ is the set of false triples generated for each training triple. The negative triple generation will be introduced in next section. P is the set of parameters to be learned in ProjR. $\lambda \sum_{\mathbf{p}\in P} \|\mathbf{p}\|$ is a regularization term with the summation of L1 norm of all elements in P. λ is the regularization parameter. The training goal is to minimize loss function L.

4 Experiments

In this paper, we conduct the experiment of link prediction and evaluate the embedding results with the benchmark knowledge graphs WN18 and FB15k which are subsets of WordNet [13] and Freebase [2] respectively (Table 3).

Table 3. Statistic of experiment dataset

Dataset	# Rel	# Ent	# Train	# Valid	# Test
WN18	18	40943	141442	5000	5000
FB15k	1345	14951	483142	50000	59071

4.1 Link Prediction

Link prediction aims to predict the missing entity given one entity and relation such as $(h, r, ?)$ and $(?, r, t)$. $(h, r, ?)$ is tail entity prediction given head entity and relation. $(?, r, t)$ is head entity prediction given tail entity and relation.

Data Prepare: As ProjR always predicts tail entities given head entity-relation pair, we regard head entity prediction $(?, r, t)$ as a tail entity prediction $(t, r^{-1}, ?)$. r^{-1} denotes the reverse relation of r. To get the embedding of r^{-1}, for each triple (h, r, t), we add reverse relation triple (t, r^{-1}, h) into training dataset.

To generate negative triples (h', r', t'), we follow the process of ProjE and randomly select m percent of entities in dataset to replace the tail entity t of (h, r, t) which means there will be $m \times n_e$ negative triples for every training triple. n_e is the number of entities in experiment dataset. $m \in (0, 1)$ is a hyperparameter.

Table 4. Results on WN18 and FB15k for link prediction. The result numbers underlined are the best results among TransR, ProjE, and ProjR. The bold result numbers are the best results among all the methods.

Method	WN18				FB15k			
	Mean Rank		Hit@10(%)		Mean Rank		Hit@10(%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
Unstructured [3]	315	304	35.3	38.2	1074	979	4.5	6.3
RESCAL [15]	1180	1163	37.2	52.8	828	689	28.4	44.1
SE [5]	1011	985	68.5	80.5	273	162	28.8	39.8
SME(linear) [3]	545	533	65.1	74.1	274	154	30.7	40.8
SME(Bilinear) [3]	526	509	54.7	61.3	284	158	31.3	41.3
LFM [8]	469	456	71.4	81.6	283	164	26.0	33.1
TransE [4]	263	251	75.4	89.2	243	125	34.9	47.1
TransH(unif) [21]	318	303	75.4	86.7	211	84	42.5	58.5
TransH(bern) [21]	401	388	73.0	82.3	212	87	45.7	64.4
CTransR(unif) [12]	243	230	78.9	92.3	233	82	44	66.3
CTransR(bern) [12]	231	218	79.4	92.3	199	75	48.4	70.2
TransD(unif) [9]	242	229	79.2	92.5	211	67	49.4	74.2
TransD(bern) [9]	**224**	**212**	79.6	92.2	**194**	91	**53.4**	77.3
TransR(unif) [12]	$\underline{232}$	$\underline{219}$	78.3	91.7	226	78	43.8	65.5
TransR(bern) [12]	238	225	79.8	92.0	198	77	48.2	68.7
ProjE_listwise [16][a]	–	–	–	–	214	60	48.1	78.8
ProjR(this paper)	356	345	$\underline{\mathbf{82.6}}$	$\underline{\mathbf{95.0}}$	$\underline{195}$	$\underline{\mathbf{41}}$	$\underline{52.3}$	$\underline{\mathbf{83.3}}$

[a]The link prediction result of ProjE on FB15k is the latest result provided by author after fixing a bug in the original code. The corresponding parameter setting of the results are: embedding dimension $d = 200$, batchsize $b = 512$, learning rate $r = 0.0005$, negative candidate sampling proportion $m = 0.1$ and max iteration number $iter = 50$.

Training: We use Adaptive Moment Estimation (Adam) [10] as the optimizer during training with the default parameter setting: $\beta_1 = 0.9, \beta_2 = 0.999$, $\epsilon = 1e^{-8}$. We also apply a dropout [18] layer on the top of combination operation to prevent overfitting and the hyperparameter of dropout rate is set to 0.5. Before training, we randomly initialize all the entity and relation vectors from a uniform distribution $U[-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}}]$ as suggested in [7]. The diagonal matrices are initialized with identity diagonal matrix. The bias vector is initialized as zero vector. For both datasets, we set the max training iterations to 100.

Evaluation: We evaluate link prediction following the same protocol of previous work: for every testing triple (h, r, t), we first predict t with input (h, r), then predict h with input (t, r^{-1}). To predict t, we replace t with each entity e in experiment dataset and calculate the similarity score through $S(h, r, e)$. Then rank the scores in ascending order and get the rank of the original right tail entity. The processing of head prediction is the same as tail prediction. Aggregating all the ranks of testing triples, we follow the two metrics used in previous work: *Mean Rank* and *Hit@10*. *Mean Rank* is the averaged rank of all the testing triples. *Hit@10* is the proportion of ranking score of testing triple that is not larger than 10. A good link predictor should achieve lower mean rank and higher hit@10. We also follow the *Filter* and *Raw* settings as previous work. *Filter* setting means filtering the triples in training data when generating negative triples to prevent false negative ones. *Raw* setting means without filtering.

Table 5. Experimental results on FB15k by mapping different patterns (%)

Method	Predict Head(Hit@10)				Predict Tail(Hit@10)			
	1-1	1-N	N-1	N-N	1-1	1-N	N-1	N-N
Unstructured [3]	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
SE [5]	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME(linear) [3]	35.1	53.7	19.0	40.3	32.7	14.9	61.6	43.3
SME(Bilinear) [3]	30.9	69.6	19.9	38.6	28.2	13.1	76.0	41.8
TransE [4]	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH(unif) [21]	66.7	81.7	30.2	57.4	63.7	30.1	83.2	60.8
TransH(bern) [21]	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
CTransR(unif) [12]	78.6	77.8	36.4	68.0	77.4	37.8	78.0	70.3
CTransR(bern) [12]	81.5	89.0	34.7	71.2	80.8	38.6	90.1	73.8
TransD(unif) [9]	80.7	85.8	47.1	75.6	80.0	54.5	80.7	77.9
TransD(bern) [9]	86.1	**95.5**	39.8	78.5	85.4	50.6	94.4	81.2
TransR(unif) [12]	76.9	77.9	38.1	66.9	76.2	38.4	76.2	69.1
TransR(bern) [12]	78.8	89.2	34.1	69.2	79.2	37.4	90.4	72.1
ProjE_listwise [16]	61.4	90.1	53.2	<u>**83.3**</u>	61.3	63.5	89.4	<u>**85.5**</u>
ProjR(this paper)	<u>**90.3**</u>	<u>93.3</u>	<u>**64.5**</u>	81.0	<u>**90.7**</u>	<u>**78.3**</u>	<u>**96.6**</u>	85.0

Result: The result of link prediction is in Table 4. We directly copy the results of previous methods from their original paper as WN18 and FB15k are two benchmark datasets. The parameter settings for ProjR's results on WN18 in Table 4 are: embedding dimension $d = 100$, learning rate $r = 0.01$, batchsize $b = 2000$, negative candidate sampling proportion $m = 0.001$. The parameter settings on FB15k are: $d = 200, r = 0.01, b = 4000, m = 0.005$. From Table 4, we could conclude that: (1) As a combination method of TransR and ProjE, ProjR achieves better performance on FB15k than ProjE and TransR. And on WN18, ProjR performs better than TransR on Hit@10. (2) Though on some tasks TransD performs better than ProjR, ProjR has less parameters than TransD. The number of parameters in ProjR is $(n_e + 4n_r + 1)k$ while that in TransD is $2(n_e + n_r)$ and $n_e \gg n_r$.

To analyze how diverse representations improve the link prediction results of different types of relations. We also conduct the experiment of link prediction

Table 6. Hit@10 of ProjE and ProjR on some examples of 8 kinds of substructures. The results are in the form of x/y in which x is the result of ProjE and y is the result of ProjR. The third column is the number of testing triples related with current row relation in testing data.

Type	Example relations	#	Hit@10 (%)	
			Head	Tail
1-1	*/influence/peer_relationship/peers*	21	66.7/**90.5**	66.7/**85.7**
	/business/employment_tenure/person	43	72.1/**79.1**	67.4/**72.1**
	/tv/tv_program/program_creator	23	**95.6/95.6**	**95.7/95.7**
1-N	*/film/writer/film*	105	92.4/**93.3**	86.7/**87.6**
	/location/country/second_level_divisions	68	**100.0/100.0**	28.4/**77.6**
	/people/cause_of_death/people	98	**87.8**/83.7	77.6/**79.6**
N-1	*/music/group_member/membership*	22	50.0/**59.1**	68.2/**77.3**
	/people/person/nationality	508	2.2/**13.2**	87.4/**93.1**
	/people/person/education/institution	358	52.2/**73.2**	69.3/**79.1**
N-N	*/award/award_winner/awards_won*	1045	90.4/**93.8**	90.4/**93.0**
	/tv/tv_genre/programs	105	**96.2**/95.2	87.6/**90.5**
	/music/genre/parent_genre	100	75.0/**83.0**	86.0/**90.0**
1-N-1	*/award/award_winner/award*	655	63.8/**64.7**	94.4/**94.7**
	/award/award_nominee/award_nominations	1555	**83.0**/82.7	94.0/**96.4**
C1	*/education/educational_institution/campuses*	60	78.3/**100.0**	80.0/**100.0**
	/education/educational_institution_campus	68	80.9/**100.0**	80.9/**100.0**
	/location/hud_county_place/place	48	20.8/**100.0**	22.9/**100.0**
C2	*/people/person/spouse_s*	54	33.3/**40.7**	27.8/**35.2**
	/influence/influence_node/influenced	235	**71.5**/63.4	**68.5**/55.7
	/location/location/adjoin_s	284	77.8/**88.7**	75.0/**83.4**
C3	*/location/location/contains*	608	92.6/**97.2**	**85.7**/84.0
	/location/adjoining_relationship/adjoins	284	77.8/**88.7**	75.0/**83.5**

mapping different structures. In this experiment, we only consider 1-1, 1-N, N-1 and N-N as the other four relation properties could be included in this four types. We choose FB15k as the dataset of this experiment because it contains more relations than WN18. The Hit@10 result of filter setting on FB15k mapping different substructures are showed in Table 5.

The results in Table 5 show that ProjR improves the ability to encode 1-1, 1-N, N-1 and N-N relations with diverse representations for entities. Among all structures, the head prediction of N-1 relations and tail prediction of 1-N relations are the most difficult tasks. And ProjR achieves good improvement on these two tasks. Compared with the second best result listed in Table 5, ProjR achieves 11.3% improvement for head prediction of 1-N relations and 14.8% improvement of tail prediction for N-1 relations.

To understand the ability of ProjR to encode the diversity of knowledge graph more deeply. We select two or three relations for each type of structure. We select relations with the principle that the number of testing triples related to the relation should be larger than 20. We compare the filter Hit@10 result of ProjE and ProjR on FB15k for each selected relation. The results are listed in Table 6. The parameter settings for ProjE and ProjR are same as the parameter settings of the results in Table 4. The results show that ProjR achieves better results in the majority of the relations for each substructure. A huge improvement is achieved on C1 structure.

5 Conclusion and Future Work

In this paper, we list 8 kinds of graph substructures to explore the structure diversity of knowledge graph and propose a new embedding method ProjR to encode structure diversity more properly based on the idea of diverse representations for entities in different relation contexts and different entity positions. In link prediction experiments, ProjR achieves better results compared with the two most related methods, TransR and ProjE. We explore the results from coarse to fine to illustrate how ProjR improves the ability to encode the diversity of structures.

There are some interesting topics that we want to explore in the future: (1) as shown in Table 6, the prediction results for different relations range hugely, which means there are still different structures between those relations. (2) Knowledge graphs are dynamic in the real world and new triples are always added to them. But existing knowledge graph embedding methods can not handle the dynamic property of KG.

Acknowledgement. This work is funded by NSFC 61473260/61673338, and Supported by Alibaba-Zhejiang University Joint Institute of Frontier Technologies.

References

1. Banko, M., Ca-farella, M.J., Soderland, S., Broadhead, M., Etzioni, O.: Open information extraction from the web (2007)
2. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of SIGMOD, pp. 1247–1250 (2008)
3. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. Mach. Learn. **94**(2), 233–259 (2014)
4. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of NIPS, pp. 2787–2795 (2013)
5. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proceedings of AAAI (2011)
6. García-Durán, A., Bordes, A., Usunier, N.: Composing relationships with translations. In: Proceedings of EMNLP, pp. 286–290 (2015)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of AISTATS, pp. 249–256 (2010)
8. Jenatton, R., Roux, N.L., Bordes, A., Obozinski, G.: A latent factor model for highly multi-relational data. In: Proceddings of NIPS, pp. 3176–3184 (2012)
9. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of ACL, pp. 687–696 (2015)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR abs/1412.6980 (2014)
11. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Proceedings of EMNLP, pp. 705–714 (2015)
12. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of AAAI, pp. 2181–2187 (2015)
13. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
14. Nickel, M., Rosasco, L., Poggio, T.A.: Holographic embeddings of knowledge graphs. In: Proceedings of AAAI, pp. 1955–1961 (2016)
15. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: Proceedings of ICML, pp. 809–816 (2011)
16. Shi, B., Weninger, T.: Proje: Embedding projection for knowledge graph completion. In: Proceedings of AAAI, pp. 1236–1242 (2017)
17. Socher, R., Chen, D., Manning, C.D., Ng, A.Y.: Reasoning with neural tensor networks for knowledge base completion. In: Proceedings of NIPS, pp. 926–934 (2013)
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
19. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of WWW, pp. 697–706 (2007)
20. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph and text jointly embedding. In: Proceedings of EMNLP, pp. 1591–1601 (2014)
21. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI, pp. 1112–1119 (2014)

22. Wang, Z., Li, J.: Text-enhanced representation learning for knowledge graph. In: Proceedings of IJCAI, pp. 1293–1299 (2016)
23. Xiao, H., Huang, M., Meng, L., Zhu, X.: SSP: semantic space projection for knowledge graph embedding with text descriptions. In: Proceedings of AAAI, pp. 3104–3110 (2017)
24. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of AAAI, pp. 2659–2665 (2016)
25. Xie, R., Liu, Z., Sun, M.: Representation learning of knowledge graphs with hierarchical types. In: Proceedings of IJCAI, pp. 2965–2971 (2016)
26. Zhang, C.: DeepDive: a data management system for automatic knowledge base construction (2015)
27. Zhang, W.: Knowledge graph embedding with diversity of structures. In: Proceedings of WWW Companion, pp. 747–753 (2017)