Learning from High-Degree Entities for Knowledge Graph Modeling

Tienan Zhang[1], Fangfang Liu[1(✉)], Yan Shen[1], Honghao Gao[1,2], and Jing Duan[1]

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China
tinanoro@163.com, 1652009609@qq.com,
{ffliu,gaohonghao,duanjing}@shu.edu.cn
[2] Computing Center, Shanghai University, Shanghai, China

Abstract. Knowledge base (KB) completion aims to infer missing facts based on existing ones in a KB. Many approaches firstly suppose that the constituents themselves (e.g., head, tail entity and relation) of a fact meet some formulas and then minimize the loss of formula to obtain the feature vectors of entities and relations. Due to the sparsity of KB, some methods also take into consideration the indirect relations between entities. However, indirect relations further widen the differences of training times of high-degree entities (entities linking by many relations) and low-degree entities. This results in underfitting of low-degree entities. In this paper, we propose the path-based TransE with aggregation (*PTransE-ag*) to fine-tune the feature vector of an entity by comparing it to its related entities that linked by the same relations. In this way, low-degree entities can draw useful information from high-degree entities to directly adjust their representations. Conversely, the overfitting of high-degree entities can be relieved. Extensive experiments carried on the real world dataset show our method can define entities more accurately, and inferring is more effectively than in previous methods.

Keywords: KB completion · Entity degree · Indirect relation · Entity prediction · Relation weakening

1 Introduction

Knowledge bases (KBs), such as Freebase [1], WordNet [2] and DBPedia [3] have recently grown in popularity since they are useful for various tasks, including information extraction [4], semantic parsing [5] and question answering [6]. These KBs contain large collections of facts about things, people and places that mostly in the form of triples (e.g., (*Bill Gates*, *FounderOf*, *Microsoft*)). A KB can be encoded as a graph, as shown in Fig. 1(a), where the nodes and edges represent entities and relations, respectively. While these KBs have been very large, they still miss large percentages of facts about common or popular entities. The lack of enough facts makes these KBs difficult to fulfill their potentials. However,

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 504–517, 2019.
https://doi.org/10.1007/978-3-030-12981-1_35

manually enriching KBs with all possible facts is impossible. Thus, researchers have devised techniques to automatically fill in missing facts by examining the facts already in the KB, which is formally known as KB completion.

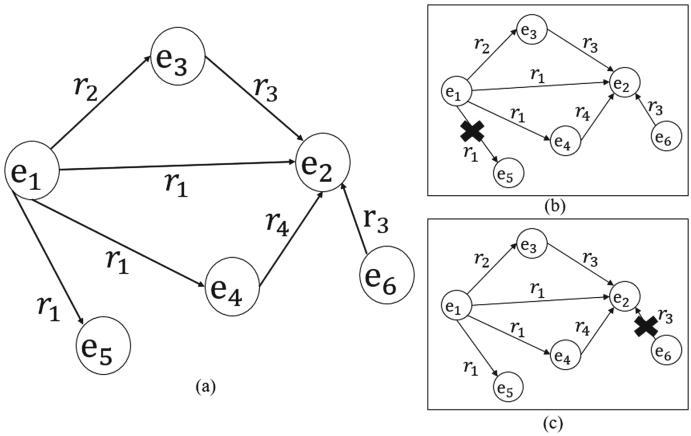

Fig. 1. (a) shows the origin knowledge base G in the form of graph; (b) demonstrates result of modeling G with PTransE, where correlation between e_1 and e_5 is weakened, even eliminated; (c) demonstrates another result that correlation between e_2 and e_6 is eliminated.

Latent feature models [7,8] are popular for performing knowledge base completion, which embed entities and relations of a KB into a continuous vector space. TransE [7] is a typical latent feature model that represents a relation with a translation vector so that the pair of embedded entities in a triple (h, r, t) (h, r and t are the head entity, relation and tail entity respectively.) can be connected by $\mathbf{r}$ with a low error. However, because of KB sparsity, the number of direct relations is small. Only utilizing the direct relations, TransE may tend to make the representations of two different entities become the same one. PTransE [9] incorporates into TransE the idea from graph feature models [10] that the relation between entities can be inferred by the indirect relations (i.e., sequences of relations) between them. In the following, direct relations and indirect relations are both referred to as path. Hence, PTransE treats each path as correlation between entities. Since high-degree entities are linked by more paths, their vector representations can be trained jointly with many entities and relations. Meanwhile, indirect relations do not bring equivalent number of paths to low-degree entities. This causes the training proportion of low-degree entities to be smaller. In addition, since vectors of entities and relations meet some formulas, the vectors of relations are also affected by the high-degree entities. Take entities e_1, e_2 and e_5 in Fig. 1(a) for example. There are three paths from e_1 to e_2: r_1, $r_2 \rightarrow r_3$ and $r_1 \rightarrow r_4$ and only one from e_1 to e_5: r_1. The quantitative differences on the paths make the correlation between e_1 and e_2 stronger and the

correlation between e_1 and e_5 weaker. In extreme cases, the correlation between e_1 and e_5 will completely disappear. As shown in Fig. 1(b), resulting model of the KB in Fig. 1(a) may lose the correlation between e_1 and e_5. Similarly, in term of e_2, e_3, e_6 and r_3, the correlation between e_2 and e_6 may also be weakened.

Consider entity a being linked by relation r to entity b and c, namely, two triples (a, r, b) and (a, r, c). In terms of basic translation model, the smaller the difference between vectors of b and c is, the less information the two triples lose. Motivated by this raw idea, we propose PTransE-ag to adjust the vector representations of related entities like the above. In fact, the nature of our method is to let entities directly learn information from each other. The effectiveness of PTransE-ag is verified on one real-world, large-scale KB: Freebase. The experimental results show that PTransE-ag substantially outperforms the baselines on missing entity prediction.

In the remainder of this paper, PTransE-ag and its implementation details are discussed in Sect. 2. The experiments and analyses are listed in Sect. 3. The related works are discussed in Sect. 4. Section 5 presents the conclusion and plans for future work.

2 Our Approach

Our method is based on PTransE [9], and utilizes its many formulas. So, we introduce PTransE briefly in Sect. 2.1. PTransE-ag will be presented in Sect. 2.2. Before proceeding, let us define our mathematical notation. We denote the knowledge base as $G = (E, R, S)$, where $E = \{e_1, \cdots, e_{|E|}\}$ is the set of entities composed of $|E|$ different entities. $R = \{r_1, r_2, \cdots, r_{|R|}\}$ is set of relations composed of $|R|$ different relations, and $S \subseteq E \times R \times E$ is the set of triples in the knowledge base.

2.1 Background

PTransE [9] broadens the correlations between entities by implementing path based translation. Path Ranking Algorithm [10] assembling sequential relations to get a path proves effective. For example, triples *(A, ParentOf, B)* and *(B, ParentOf, C)* can form triple *(A, [ParentOf, ParentOf], C)*. *[ParentOf, ParentOf]* corresponds to *GrandparentOf*. Obviously, composed relations can reflect the correlation between *A* and *C*. Hence, PTransE utilizes some paths for each pair of entities to strengthen their correlations. Firstly, paths whose lengths are less than 3 are preserved. Then, path-constraint resource allocation algorithm (PCRA) computes the reliability of relation paths. Finally, reliable paths are selected for representation learning.

PCRA associates a certain amount of resources with the head entity h and then distributes resource along the given path p. The resource that eventually flows to the tail entity t is the reliability of the path p, which is denoted as $R(p|h, t)$. The number of resources flowing to t is defined in [9] as follows:

$$R_p(t) = \sum_{n \in S_{i-1}(\cdot,t)} \frac{1}{|S_i(n, \cdot)|} R_p(n), \tag{1}$$

where $S_i(n, \cdot)$ is the direct successors of $n \in S_{i-1}$, following the relation r_i, and $R_p(n)$ is the number of resources flowing to entity n.

Composition methods of relations include: ADD, MUL and RNN. ADD is an addition operation $p = r_1 + r_2 + \cdots + r_l$. MUL is cross product $p = r_1 \times r_2 \times \cdots \times r_l$. RNN refers to recurrent neural networks. Since its performance is weak, our method will not use it.

Following TransE, PTransE defines energy function for a multi-relation path triple (h, p, t) as $E(h, p, t) = ||h + p - t||$. Since $||h + r - t||$ has been minimized to make sure $r \approx t - h$, the loss function of (h, p, t) can be transformed to $E(h, p, t) = ||p - (t - h)|| = ||p - r|| = E(p, r)$, which is expected to be low when the multiple-relation path p is consistent with the direct relation.

2.2 PTransE-ag

As shown in Fig. 1(a), there are three paths from e_1 to e_2: r_1, $r_2 \rightarrow r_3$ and $r_1 \rightarrow r_4$ and only one from e_1 to e_5: r_1. The quantitative differences of paths make the correlation between e_1 and e_2 stronger and the correlation between e_1 and e_5 weaker. In extreme cases, the correlation between e_1 and e_5 will completely disappear. The resulting model of the KB in Fig. 1(a) may become Fig. 1(b). In a word, the relations between low-degree entities and other entities are weakened. We call this problem the *relation weakening problem*.

To address the relation weakening problem, we utilize the fact that two entities being both related to another entity by the same relation means that these two entities have some similarities. Given three entities a, b and c, if a is related to b by relation r and b is similar to c, then there also should be relation r between a and c. Figure 2(a) demonstrates the weakening of the relation between e_1 and e_5, where the entities and relations are represented as vectors to better display the translation. The result of the translation on e_1 is far away from e_5. To make e_5 closer to $e_1 + r_1$, our method reduces the distance between e_2 and e_5 (as shown in Fig. 2(b)), which is equivalent to increasing their similarities. Since our method corresponds to forming aggregations of entities and is based on PTransE, it is named as PTransE-ag. In addition, experiments show that directly applying aggregation on TransE would not improve the efficiency. This suggests that considering the similarities between entities modeled by few correlation information makes no sense.

Optimization Objective. Note that we will not judge whether some relations between entities are weakened. Instead, we construct a valid triple during the training of triple (h, r, t), and then directly increase the similarity between the component and its replacement to reach the goal of strengthening relations. For the sake of understanding, the valid triple is denoted as (h, r, t') which means that it is constructed by replacing t by t'. PTransE-ag aims to ensure that the loss of (h, r, t') will not be too large when minimizing the loss of (h, r, t). The larger the loss of (h, r, t') is, the weaker the correlation between h and t' is. Because the only difference between (h, r, t) and (h, r, t') is the tail entity and

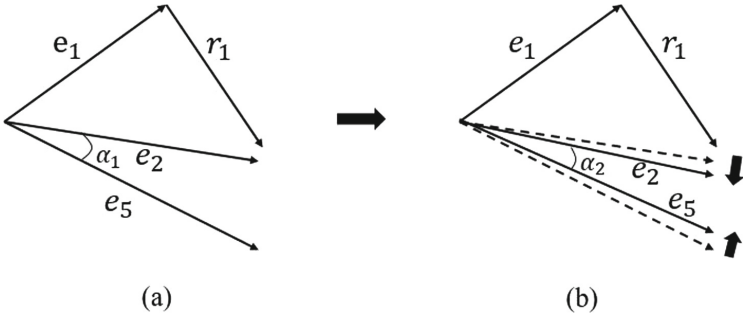

Fig. 2. (a) illustrates the outcome of modeling two triples (e_1, r_1, e_2) and (e_1, r_1, e_5), where e_5 is too far away from $e_1 + r_1$. (b) demonstrates the method to make e_5 closer to $e_1 + r_1$, i.e., reducing the distance between them ($\alpha_2 < \alpha_1$).

the loss of (h, r, t) has been minimized, the loss of (h, r, t'') can be reduced by reducing the distance between t and t''. In short, PTransE-ag balances the losses of (h, r, t) and (h, r, t'') to reduce the global loss of the KB. The optimization objective of PTransE-ag is defined as

$$L(S) = \sum_{(h,r,t)\in S} [L(h, r, t) + L_1(h, r, t) + \frac{1}{Z} \sum_{p\in P(h,t)} R(p|h, t)L(p, r)], \quad (2)$$

where components except $L_1(h, r, t)$ are defined in PTransE [9]. Therefore, we just list their definitions without detailed explanation. $Z = \sum_{p\in P(h,t)} R(p|h, t)$ is a normalization factor, and $P(h, t)$ is the paths set between h and t. Following TransE, $L(h, r, t)$ and $L(p, r)$ are loss functions with respect to the triple (h, r, t) and the pair (p, r):

$$L(h, r, t) = \sum_{(h',r',t')\in S'} [\gamma + E(h, r, t) - E(h', r', t')]_+, \quad (3)$$

and

$$L(p, r) = \sum_{(h,r',t)\in S'} [\gamma + E(p, r) - E(p, r')]_+. \quad (4)$$

where $[x]_+ = max(0, x)$ returns the maximum between 0 and x, γ is the margin, S is the set of valid triples in KB. S' is the set of invalid triples. $E(h, r, t) = ||h+r-t||_1$ and $E(p, r) = ||p-r||_1$ are energy functions. And Formula $L_1(h, r, t)$ is defined as

$$L_1(h, r, t) = \begin{cases} 0, ||h - h''||^2 < \beta \text{ and } ||t - t''||^2 < \beta \text{ and } ||r - r''||^2 < \beta \\ \sum\limits_{(h'',r'',t'')\in S''} ||h - h''||^2 + ||r - r''||^2 + ||t - t''||^2, \text{otherwise} \end{cases} \quad (5)$$

$\beta > 0$ is the threshold of the distance between the entities or relations. S'' is the set of valid triples constructed by replacing one of the three components

of a triple. In Eq. (5), the error does not exist when the distances between the corresponding components of (h, r, t) and (h'', r'', t'') are all less than β (Note that there always be only one difference between (h, r, t) and (h'', r'', t'').). We use Eq. (5) to realize the aggregation. Here, the aggregation is extended to relation. We think that if two relations link the same head and tail entity, then they should also have some similarities. If the distance between a component and its replacement exceeds β, the distance between them should be reduced, such as with e_2 and e_5 in Fig. 1(a). In fact, our method adds the term $L_1(h, r, t)$ into PTransE optimization objective.

Implementation Detail. Before proceeding, the idea of the margin-based loss is worth discussing. Ideally, the loss of each triple is expected as small as possible. But explicitly specifying how small the loss should be is infeasible. Hence, the margin-based loss is employed to make the loss of invalid triple at least one margin larger than the loss of valid triple. Only when the loss difference is less than margin does the error exist, and then, parameter updating is executed.

Algorithm 1. Learning PTransE-ag

Input: Training set $S_{(h,r,t)}$, entities and rel. sets E and L; path sets P; path reliability set R; learning rate λ, margin γ, embeddings dim. k, threshold β.

1: **initialize** $r \leftarrow uniform(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each relation $r \in L$,
$e \leftarrow uniform(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$ for each entity $e \in E$
2: **loop**
3: $e \leftarrow \frac{e}{||e||}$ for each $e \in E$
4: $r \leftarrow \frac{r}{||r||}$ for each $r \in L$
5: $S_{batch} \leftarrow sample(S, b)$ //sample a minibatch of size b
6: $T_{batch} \leftarrow \emptyset$ //initialize the set of triplets of triples
7: **for** $(h, r, t) \in S_{batch}$ **do**
8: $(h', r', t') \leftarrow sample(S'_{(h,r,t)})$ //sample a invalid triple
9: $(h'', r'', t'') \leftarrow sample(S''_{(h,r,t)})$ //sample a valid triple
10: $T_{batch} \leftarrow T_{batch} \bigcup \{((h, r, t), (h', r', t'), (h'', r'', t''))\}$
11: **end for**
12: Update embeddings w.r.t.
13: $\sum_{((h,r,t),(h',r',t'),(h'',r'',t'')) \in T_{batch}} [\nabla L(h, r, t) + \nabla L_1(h, r, t) + \frac{1}{Z} \sum_{p \in P(h,t)} R(p|h, t) \nabla L(p, r)]$
14: **end loop**

The detailed optimization procedure is described in Algorithm 1. Algorithm 1 is similar to that of PTransE [9], with the only difference that PTransE-ag needs an extra valid triple set, denoted as S''(Line 9, Algorithm 1), to reduce the distance between similar entities. For optimization, we employ the stochastic gradient descent (SGD) with constant learning rate to minimize the loss function. All embeddings for entities and relations are first initialized following the random procedure proposed in [11]. At each main iteration of the algorithm, the

embeddings are first normalized. Then, a small set of triples is sampled from the training set, and will serve as the training samples of the minibatch. For each training sample (h, r, t), randomly select a component to be replaced. For the sake of description, we take t as example. Then, the updating (step 13 in Algorithm 1) performs the following steps:

Step-1. Construct a invalid triple (h, r, t'). If **Condition-1** (The loss of training sample (h, r, t) is not at least one margin smaller than the loss of invalid triple (h, r, t').) is met, the embeddings of h, r, t and t' will be updated.
Step-2. Construct a valid triple (h, r, t''). Note that (h, r, t'') may not exist. If **Condition-2** ((h, r, t'') exists and the distance between the embeddings of t and t'' exceeds threshold β) is met, the embeddings of t and t'' will be updated. This step is unique to PTransE-ag.
Step-3. Construct a invalid triple (h, r', t). This step involves the path-based translation. Like **Step-1**, for each path p between h and t, if **Condition-3** (The loss of (p, r) is not at least one margin smaller than the loss of (p, r').) is met, the embeddings of r, r' and relations in path p will be updated. Note that no matter which component in training sample (h, r, t) is replaced, Step-3 always replaces r of (h, r, t). This is because the path-based translation only uses relation and path.

Among three steps above, **Step-2** is our work integrated into PTransE. Therefore, the difference between PTransE-ag and PTransE is that PTransE-ag will take into account other valid triples when training a triple.

Evaluation Function. The evaluation function of PTransE is directly used as ours. In TransE, the loss function of a triple also works as its evaluation function. Similarly, the evaluation function of PTransE-ag can also be derived from its loss function. It is defined as

$$S(h, r, t) = G(h, r, t) + G(t, r^{-1}, h), \tag{6}$$

where $G(h, r, t)$ is defined as

$$G(h, r, t) = E(h, r, t) + \frac{1}{Z} \sum_{p \in P(h,t)} Pr(r|p) R(p|h, t) E(p, r). \tag{7}$$

The inversion of the relation (e.g., r^{-1}) is needed in paths and learned in the path-based translation. $Pr(r|p) = \frac{Pr(r,p)}{Pr(p)}$ is the global correlation between r and p, where $Pr(x)$ is the number of x in the KB.

3 Experiments and Analysis

In this section, experiments on *entity prediction* is performed. Given an entity e and a relation r, *entity prediction* discovers the entities which are most likely to be related to e by relation r.

Dataset. To make a direct comparison to PTransE [9], we evaluate our method on the dataset used in PTransE, that is, a dataset extracted from Freebase, FB15K [7]. The statistics of FB15K are listed in Table 1.

Table 1. Statistics of dataset

Dataset	#Relation	#Entity	#Train	#Valid	#Test
FB15k	1,345	14,951	483,142	50,000	59,071

Evaluation Metrics. For each testing triple (a.k.a., seeded triple), we construct a set of triples by replacing one component of seeded triple and then compute scores of these triples including seeded triple with evaluation function. These triples are sorted in ascending order to evaluate the performance of an approach. Here, we concentrate on two evaluation metrics [7]: $Hits@n$ and $Mean$. $Hits@n$ records the proportion of seeded triples ranking in top n and $Mean$ records the mean of seeded triples' ranks. For $Mean$, the smaller the figure is, the better the performance of a method is. For $Hits@n$, it is quite the contrary. These two metrics are defined as follows:

$$Hits@n = \frac{num\ of\ seeded\ triples\ ranking\ in\ top\ n}{num\ of\ seeded\ triples} \tag{8}$$

$$Mean = \frac{\sum_{triple\ in\ testing\ set} rank\ of\ triple}{num\ of\ seeded\ triples} \tag{9}$$

However, the method of replacing one component is flawed when a constructed triple that is not in testing set ends up being valid in the KB. For example, one of our testing triples is (*NewYork*, *locatedIn*, *USA*). Then, during prediction, the triple (*Chicago*, *locatedIn*, *USA*) is constructed, which is obviously not the original testing triple but is also valid in the KB. This kind of triples will be regarded as invalid because in the evaluation only the ranks of the seeded triples are the concern. This is not reasonable. Therefore, it is necessary to filter out those undesirable valid triples during the evaluation. We name the raw evaluation setting mentioned before as *Raw*, the filtering setting as *Filter*.

Baselines. Although many recent works are referred to in Sect. 4, we only pick some early variants of TransE as our baselines because the aim of comparison is to verify if our modification is efficient. The employed baselines include SE [12], TransE [7], TransH [8], TransR [13] and PTransE [9].

Parameters Configurations. The optimal configurations of PTransE-ag are $\lambda = 0.001$, $\gamma = 1$, $k = 100$, and $\beta = 0.05$ and we take L_1 as dissimilarity of the loss function. λ is the learning rate of SGD, γ is the margin between loss of a valid triple and the loss of its corresponding invalid one, k is the dimensionality

of entity and relation embeddings and β is the threshold of the distance between entities or relations. The number of training epochs over all training triples is set to 500.

Entity Prediction. In entity prediction, the missing head or tail entity of a triple is inferred. Given a seeded triple, construct triples by replacing tail entity by each member of entity set E. Because searching path is of high memory-consumption, we first employ evaluation function of TransE to select newly constructed triples ranking in top 500. These top 500 triples are then sorted according to scores computed by evaluation function of PTransE-ag in ascending order. Finally, record the rank of the seeded triple (Usually, seeded triple is in top 500). Having iterated over all seeded triples, we can compute values of $Hits@n$ and $Mean$.

Experiment Analysis. Results on entity prediction are demonstrated in Table 2, where four metrics are listed: the *Raw* and *Filter* version of $Mean$ and $Hits@10$. For PTransE, we list its results with different relation composition operations and step length. For example, PTransE(ADD,2-step) means that the composition operation is addition and the length of the step is not more than 2. For PTransE-ag, we only list the results of the ADD and MUL composition operation because RNN works worse than the other two composition operations in PTransE. Moreover, due to the complexity of the cross product of vectors, only (ADD,3-step) is listed. Aggregation is also applied on TransE, which is TransE-ag in Table 2. TransE-ag's performance is worse than TransE's. This is because TransE only takes into consideration the direct relations between entities. Direct relations are not enough to model an entity, let alone to judge if two entities have similarities. The best result for each metric is shown in bold font.

Table 2 shows that PTransE-ag consistently outperforms baselines including PTransE. This indicates that preventing the information of direct relations from being overwhelmed by paths can better define an entity. No matter whether PTransE or PTransE-ag is used, (ADD,2-step) always outperforms its counterparts on most of the metrics. This indicates that the addition composition operation is effective and too long paths provide little information on entities. In addition, PTransE with RNN obtains the worst performance among three kinds of composition operations. This suggests that complex models unnecessarily outperform simple models.

Tables 3 and 4 demonstrate the performance of PTransE and some baselines with respect to different types of relations. Relations can be categorized into four classes according to the cardinalities of their head and tail arguments: 1-to-1, 1-to-N, N-to-1 and N-to-N. On all types of relations, PTransE-ag achieves the best results. It appears that predicting entities on the "side 1" is easier (e.g., predicting the head entities in 1-to-N relations and tail entities in N-to-1 relations). However, in the other hand, it can be observed that N-to-1 relations in head entity prediction and 1-to-N relations in tail entity prediction always obtain the worst performance in all methods and they perform much worse than other

Table 2. Evaluation results on entity prediction

Metric	Mean rank		Hits@10(%)	
	Raw	Filter	Raw	Filter
SE	273	162	28.8	39.8
TransE	243	125	34.9	47.1
TransE-ag	252	133	33.2	45.9
TransH	212	87	45.7	64.4
TransR	198	77	48.2	68.7
PTransE(ADD, 2-step)	200	54	51.8	83.4
PTransE(MUL, 2-step)	216	67	47.4	77.7
PTransE(RNN, 2-step)	242	92	50.6	82.8
PTransE(ADD, 3-step)	207	58	51.4	84.6
PTransE-ag(ADD, 2-step)	182	**45**	**53.9**	**87.8**
PTransE-ag(MUL, 2-step)	205	56	52.7	87.7
PTransE-ag(ADD, 3-step)	**180**	46	53.2	82.1

Table 3. Head prediction by mapping properties of relations. (%)

Tasks	Predicting head entities(Hits@10)			
Relation Category	1-to-1	1-to-N	N-to-1	N-to-N
SE	35.6	62.6	17.2	37.5
TransE	43.7	65.7	18.2	47.2
TransH	66.8	87.6	28.7	64.5
TransR	78.8	89.2	34.1	69.2
PTransE(ADD, 2-step)	91.0	92.8	60.9	83.8
PTransE(MUL, 2-step)	89.0	86.8	57.6	79.8
PTransE(RNN, 2-step)	88.9	84.0	56.3	84.5
PTransE(ADD, 3-step)	90.1	92.0	58.7	86.1
PTransE-ag(ADD, 2-step)	91.4	**96.3**	**64.2**	88.9
PTransE-ag(MUL, 2-step)	**92.6**	95.7	63.4	**89.4**
PTransE-ag(ADD, 3-step)	90.0	94.6	58.0	82.2

types of relations. This reflects the inherent problem of TransE-series methods that non-1-to-1 relations are incompatible with the translation operation.

4 Related Work

The task of knowledge base completion has seen a lot of attention in recent years. Statistical relational learning (SRL) is widely applied on KB completion. We will touch on the latent feature models. Approaches of latent feature

Table 4. Tail prediction by mapping properties of relations. (%)

Tasks	Predicting tail entities(Hits@10)			
Relation Category	1-to-1	1-to-N	N-to-1	N-to-N
SE	34.9	14.6	68.3	41.3
TransE	43.7	19.7	66.7	50.0
TransH	65.5	39.8	83.3	67.2
TransR	79.2	37.4	90.4	72.1
PTransE(ADD, 2-step)	91.2	74.0	88.9	86.4
PTransE(MUL, 2-step)	87.8	71.4	72.2	80.4
PTransE(RNN, 2-step)	88.8	68.4	81.5	86.7
PTransE(ADD, 3-step)	90.7	70.7	87.5	88.7
PTransE-ag(ADD, 2-step)	90.1	74.2	**94.8**	**90.7**
PTransE-ag(MUL, 2-step)	**92.0**	**74.3**	92.7	90.7
PTransE-ag(ADD, 3-step)	89.2	67.3	93.5	85.0

models usually learn an embedded representation of entities and relations in low-dimensional spaces to capture the correlation between the entities/relations using latent variables. Our approach PTransE-ag also fits in this line of work.

Some earliest works include collective matrix factorization models represented by RESCAL [14] and energy-based frameworks represented by SE [12]. However, these methods acquire great expressivity at the expense of substantial increases in model complexity.

To achieve better trade-offs between accuracy and scalability, TransE [7] represented each entity with a constant vector regardless of the relation type linking this entity. TransH [8] and TransR [13] argued that an entity should behave diversely when linking different relations, and thus proposed to transform the representation of an entity based on current relation before translation. TranSparse [15] proposed to substitute sparse matrices for dense matrices of TransR to deal with the heterogeneity and the imbalance of KB. TransG [16] assumed that each relation corresponds to various semantics, each of which can be revealed by the entity pairs associated with triples. KR-EAR [17] distinguished relation types and classified them into two classes, i.e., relations and attributes. The former indicates the relationship between entities. The other one represents the properties of entities. Similar methods that tried to discovered geometric structure of the embedding space also include TransD [18], TransA [19] and TransCoRe [20]. TransD learned projection matrix by considering the interaction among relations and entities. TransA introduced Mahalanobis distance into the energy function of TransE because original L_1 or L_2 distance is sort of simple and treats each dimension of representations equally. TransCoRe further employed SVD to find out that a small number of dimensions can capture most information of a relation, and thus converted the problem of learning the

embedded relation matrix into learning two low-dimensional matrices. In this way, each relation can be represented by less dimensions with a shared basis.

Although innovative and efficient, above methods only tried to exploit more information from relation-based triples. In fact, relation paths can also reflect the correlations among entities and relations. PTransE [9] learned the vector representations of paths to enrich the number of triples and thus obtained high performance on knowledge base completion. DPTransE [21] computed the contribution of paths to a certain relation by employing a graph feature model, and then learned the representations of knowledge base based on a modified TransE model. However, to obtain the contribution of paths, DPTransE needed predefined embeddings of relations in KB. RPE [22] embedded entities and relations into different low-dimensional spaces with semantics of relation paths. That is, RPE simultaneously embedded each entity into two types of latent spaces. Other works will also consider logic rules. TARE [23] integrated knowledge triples and logic rules and emphasized the importance of transitivity and asymmetry of logic rules to order the relation types. All these latent feature models tried to minimize the margin-based loss. However, they ignored the problem that some other valid triples are likely to be discarded while optimizing current triple.

5 Conclusion and Future Work

Our method aims to address the problem that indirect relations weaken correlations between some entities. PTransE obtains significant performance improvement but it ignores the side effect paths bring. However, it is the side effect that guides us to the workaround. It is found that high-degree entities are trained much more than low-degree entities do, which means more information are embedded in high-degree entities. Tracking down the basic idea of translation model, we proposed PTransE-ag that allows low-degree entities to directly learn from high-degree entities. And experiments also prove the effectiveness of PTransE-ag.

In addition, there are some shortcomings of our method. Firstly, the training time is an issue due to the extra time for finding related entities. This can be improved by optimizing the searching algorithm. In fact, the running time matters the most, which remains the same as PTransE's. Secondly, the relation weakening problem is from our intuition. Although we follow this intuition and propose PTransE-ag that surely improves performance, concrete evidences still remain to be proposed. Our future work will explore more detailed experiments. Lastly, PTransE-ag simply shrinks the difference between the vector of similar entities. It overlooks the motivation of low-dimensional vector representation that the features of an entity can be distributed represented. It is noteworthy that each dimension of an entity usually plays a different role when this entity interacts with different relations. Therefore, our future work will center on the distinguishing of dimensionality. For example, SVD [24] can be employed to learn the primary component of a vector. This may guide the direction along which entities should be closed to each other. In addition, we will further extend our

method to the relation prediction task to examine and improve the universality of PTransE-ag.

Acknowledgements. This work is supported by National Key Research and Development Plan of China under Grant No. 2017YFD0400101, and National Natural Science Foundation of China under Grant No. 61502294, and Natural Science Foundation of Shanghai under Grant No. 16ZR1411200.

The url of the source code is https://github.com/IdelCoder/PTransE-ag.

References

1. Evans, C., Paritosh, P., Sturge, T., Bollacker, K., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250 (2008)
2. Miller, G.A.: WordNet: a lexical database for English. Future Gener. Comput. Syst. **38**(11), 39–41 (1995)
3. Jakob, M., Mendes, P.N., Bizer, C.: DBpedia: a multilingual cross-domain knowledge base. In: Proceedings of Language Resources and Evaluation, pp. 1813–1817 (2012)
4. Zhou, M., Nastase, V.: Using patterns in knowledge graphs for targeted information extraction. In: KBCOM 2018 (2018)
5. Gesmundo, A., Hall, K.: Projecting the knowledge graph to syntactic parsing. In: Proceedings of Conference of the European Chapter of the Association for Computational Linguistics, pp. 28–32 (2014)
6. Singh, K., Diefenbach, D., Maret, P.: WDAqua-core1: a question answering service for RDF knowledge bases. In: WWW 2018 Companion (2018)
7. Usunier, N., Garcia, A., Weston, J., Bordes, A., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of International Conference on Neural Information Processing Systems, pp. 2787–2795 (2013)
8. Zhang, J., Feng, J., Wang, Z., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 1112–1119 (2014)
9. Liu, Z., Luan, H., Sun, M., Rao, S., Lin, Y., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 705–714 (2015)
10. Mitchell, T., Lao, N., Cohen, W.W.: Random walk inference and learning in a large scale knowledge base. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, pp. 27–31 (2011)
11. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. Mach. Learn. **9**, 249–256 (2010)
12. Weston, J., Collobert, R., Bordes, A., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 301–306 (2011)
13. Liu, Z., Lin, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 2187–2195 (2015)
14. Tresp, V., Nickel, M., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Proceedings of International Conference on Machine Learning, pp. 809–816 (2011)

15. Liu, K., He, S., Ji, G., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: Proceedings of AAAI Conference on Artificial Intelligence, pp. 985–991 (2016)
16. Huang, M., Xiao, H., Zhu, X.: TransG: a generative model for knowledge graph embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 2316–2325 (2016)
17. Liu, Z., Lin, Y., Sun, M.: Knowledge representation learning with entities, attributes and relations. In: Proceedings of International Joint Conference on Artificial Intelligence, pp. 2866–2872 (2016)
18. He, S., Xu, L., Liu, K., Ji, G., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pp. 687–696 (2015)
19. Huang, M., Yu, H., Xiao, H., Zhu, X.: TransA: an adaptive approach for knowledge graph embedding (2015)
20. Jia, Y., Zhu, J., Qiao, J.: Modeling the correlations of relations for knowledge graph embedding. J. Comput. Sci. Technol. **33**(2), 323–334 (2018)
21. Wang, Q., Xu, W., Li, W., Zhang, M., Sun, S.: Discriminative path-based knowledge graph embedding for precise link prediction. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) ECIR 2018. LNCS, vol. 10772, pp. 276–288. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76941-7_21
22. Liang, Y., Giunchiglia, F., Feng, X., Lin, X., Guan, R.: Relation path embedding in knowledge graphs. Neural Comput. Appl. 1–11 (2018)
23. Rong, E., Zhuo, H., Wang, M., Zhu, H.: Embedding knowledge graphs based on transitivity and asymmetry of rules. xplan-lab.org (2018)
24. Kalman, D.: A singularly valuable decomposition: the SVD of a matrix. Coll. Math. J. **27**(1), 2–23 (1996)