Domain Representation for Knowledge Graph Embedding

Cunxiang Wang[1,2], Feiliang Ren[3], Zhichao Lin[3], Chenxu Zhao[3], Tian Xie[3] and Yue Zhang[2]

[1] College of Computer Science and Technology, Zhejiang University, China
wangcunxiang@westlake.edu.cn
[2] School of Engineering, Westlake University, China
yue.zhang@wias.org.cn
[3] School of Computer Science and Engineering, Northeastern University, China
renfeiliang@cse.neu.edu.cn, enjoymath2016@163.com
{ch4osmy7h,thankoder}@gmail.com

Abstract. Embedding entities and relations into a continuous multi-dimensional vector space have become the dominant method for knowledge graph embedding in representation learning. However, most existing models ignore to represent hierarchical knowledge, such as the similarities and dissimilarities of entities in one domain. We proposed to learn a Domain Representations over existing knowledge graph embedding models, such that entities that have similar attributes are organized into the same domain. Such hierarchical knowledge of domains can give further evidence in link prediction. Experimental results show that domain embeddings give a significant improvement over the most recent state-of-art baseline knowledge graph embedding models.

Keywords: Representation learning · Knowledge graph · Domain

1 Introduction

Containing relational knowledge between entities, Knowledge Graphs [12, 7, 9, 18] can help improve reasoning in QA systems [10], conversation systems [18] and recommendation systems [3]. Intuitively, a more comprehensive knowledge graph will be more beneficial for its applications. But knowledge graphs are far from complete [12]. Knowledge graph embedding models can be helpful for expanding knowledge graphs. The basic idea is to project entities and relations into a continuous multi-dimension space so that new relational facts can be scored for their credibility in a dense vector space. This task is called link prediction [2].

Numerous models have been proposed for knowledge graph embedding, including TransE [2], and TransR [7], which learn embeddings of entities and relations by leveraging their distributed contexts. One issue of TransE and its subsequent variants, however, is that knowledge is not organized hierarchically. For example, all geographic knowledge share common attributes, a categorical representation of which can facilitate link prediction. We address this issue by introducing the concept of *domains*, which are collections of entities around a certain relation. For example, consider the relation "Capital", for which the head entity must be a country, and the tail entity must be a capital city. In this example, the set "country" and the set "capital city" are both *domains*.

Domains add a layer of abstraction to KG embeddings. Knowledge on whether a given entity belongs to a given relation's head or tail domain is helpful for link prediction. For example, when considering two triples (USA, capital, Washington) and (USA, capital, New York) as candidate new triples, Washington D.C. can be inside the "capital city" domain while New York can be outside the domain. As a result, (USA, capital, New York) should suffer a penalty given such domain knowledge. In contrast, preliminary experiments show that when using TransR/STransE for link prediction, more than half of incorrect entities do not belong to the right domain.

We propose a model for learning explicit domain representation given a KG. Since entities in a domain are similar in some attributes, while dissimilar in other attributes, domains are restricted using hyper-ellipsoids in the vector space. Given a trained KG, we learn the representation of each domain by fitting one hyper-ellipsoid to a hyper-point cluster with similar attributes. The training objective is set to minimize the overall distance between entities and the domain surface. We approximately calculate the distance measure for a simple, concise algorithm with high runtime efficiency. The underlying KG embedding models or their embeddings do not change during training. In link prediction, we calculate the distance between each candidate entity and the target domain, adding the distance to the baseline score to rank candidate entities.

Experiments show that our model is effective over strong baseline models. To our knowledge, we are the first to explicitly learn domain representations over knowledge graph embeddings. We release our source code and models at https://github.com/wangc unxiang/Domain-Representation-for-Knowledge-Graph-Embedding.

2   Related Work

Prior knowledge graph embedding models and the works about domains are related to our model. Knowledge graph embedding models are either employed with external information or not. For those without external information, there are two main streams-*Translation-based models* and *neural network models*. Models using external information use different types of resources.

2.1   Models without Using External Information

Translation-based models treat entities as a hyper-points and relations as a vectors in the vector space. The training objectives are set to ensure certain correlations between points and vectors. TransE [2] is a seminal work of all translation-based models, it believes that head entity plus relation approximately equals tail entity in the vector space. Subsequently, TransH [14] overcomes the flaws of TransE concerning the 1-to-N/N-to-1/N-to-N relations. TransR [7] builds entity and relation embeddings in separate entity and relation spaces. TransSparse [5] aims to handle heterogeneity and imbalance of data, and STransE [9] models head and tail spaces differently. TransD [4] considers entities for projection matrices. TransA [6] makes the margin changes dynamically. TransG [15] aims to solve the problem of multiple-relation semantics, and ITransF [16] uses sparse attention to solve the problem of data sparsity.

Among *neural network models*, SLM (Single Layer Model) [12] applies the neural network to knowledge graph embedding. NTN (Neural Tensor Network) [12] uses a bilinear tensor operator to represent each relation. ProjE [11] can be seen as a modified version of NTN.

We choose translation-based models as baselines since they are more efficient and highly effective compared to neural network based models.

2.2   Models Using External Information

Text-aware models import external information. The main idea is to employ textual representation or attributes information of entities and relations to existing models (e.g. TransE), which also means that text aware models cannot work independently and have to be attached to a knowledge graph embedding model to improve the baseline model performance. In this sense, text-aware models are similar to our domain representation model. However, our model does not need any external information.

2.3   Investigation of Domains

Some research on knowledge graphs can be regarded as domain related. For example, Dual-Space Model [13] is designed for calculating the similarity between different domains and functions to help semantic similarity task, but cannot be used for link prediction. Another example, [17] utilize learned relation embeddings to mine logic rules, such as *BornInCity(a,b) ^CityOfCountry(b,c) ⇒ Nationality(a,c)*. The concept of domains is used to restrict search choices of logic rules. Though the models above use entities in domains, none of them tries to represent domains explicitly in the vector space, which is the core idea of our model. As a result, they cannot extract the common attributes of domains.

3   Baselines

In this section, we introduce the baseline models - *Translation-based models* including TransE [2], TransR [7] and STransE [9], as well as the main task of link prediction.

3.1   Three Translation-based models

TransE [2] is the root of all translation-based models. As a seminal work, given a head entity h, a relation r and a tail entity t, TransE [2] models a relation triple $\langle h, r, t \rangle$ with $h + r \approx t$. The training objective function is thus to minimize

$$f_r\left(h, t\right) = \|h + r - t\|_{l_{1/2}} \tag{1}$$

over a whole KG. Pre-trained with the head entity vectors hs, the relation vectors rs and tail entity vectors ts by TransE, TransR [7] uses one projection matrix per relation to do translational operation in the relation space, with an objective function

$$f_r\left(h, t\right) = \|W_r h + r - W_r t\|_{l_{1/2}} \tag{2}$$

where $W_r \in R^{k \times d}$ is the projection matrix. STransE [9] is similar to TransR; It also uses pre-trained entity vectors and relation vectors outputted by TransE. But for

STransE, each relation has two projection matrices, one for head entities, the other for tail entities. STransE's objective function is

$$f_r(h,t) = \|W_{r,1}h + r - W_{r,2}t\|_{l_{1/2}} \tag{3}$$

where $W_{r,1} \in R^{k \times d}$ and $W_{r,1} \in R^{k \times d}$ are the projection matrices for head entities and tail entities, respectively.

3.2   Link Prediction

Link prediction is one of the most common evaluation protocols in knowledge graph embedding. Given a test triple $< h, r, t >$, where h, r and t denote the head entity, the relation and the tail entity, respectively. The task is to predict the second entity (h or t) once the relation and one entity are determined. We first remove the head entity, replacing it with all entities in the knowledge graph to calculated a fitness score for each entity, based on the objective function of the current model. All the entities are ranked according to the score, and the rank of h among all entities is recorded. We repeat the same procedure for the tail entity. The two ranks are used for the credibility score of the triple $< h, r, t >$, which will be used in subsequent evaluation metrics.

4   Domain Representation Using Ellipsoids (DRE)

We model domain structures in the vector space as hyper-Ellipsoids. Section 3.1 introduces a formal definition of domains and explains why we use hyper-ellipsoids to represent domains. Section 3.2 discusses the formal representation of a hyper-dimensional hyper-ellipsoid. Section 3.3 describes how domains can be used for tasks related to knowledge graphs. Section 3.4 discussed training. Finally, Section 3.5 illustrates how the domain model can be used in combination with various knowledge graph embedding models.

4.1   Domain and Hyper-Ellipsoid

We define a *domain* in a knowledge graph as the set of a relation's head or tail entities. Formally, for any relation r, its head domain $D_{h,r}$ is

$$D_{h,r} = \{e_h \in E | \exists e_t \in E \land (e_h, r, e_t) \in T\} \tag{4}$$

And its tail domain $D_{t,r}$ is

$$D_{t,r} = \{e_t \in E | \exists e_h \in E \land (e_h, r, e_t) \in T\} \tag{5}$$

where T is the set of all triples, E is the set of all entities.

From our perspective, the concept of a domain is closely related to the concept of similarity. Entities in a domain have similar attributes and are close in certain dimensions in the vector space. Other the other hand, the shape of a domain in a vector space distributes unevenly in different dimensions since the head/tail entities are only similar in some attributes, but unrelated or even distinct in others. For instance, to the relation "Capital", "Beijing" and "Washington" can both be tail entities. However, they can be very different in other senses. For example, "Beijing" is a large city with more than 20M
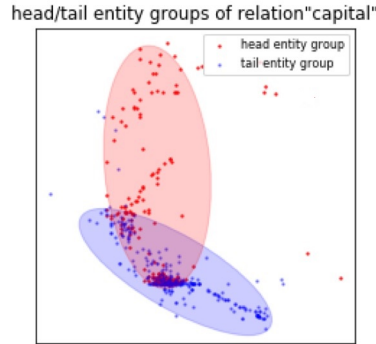
Fig. 1: One relation and its head/tail entity group, trained on FB15K by TransE (vectors not normed). Most head/tail entities are close in one dimension but scattered in other dimensions respectively.

people while "Washington" is a small city with only 700K people, and they are also far away from each other in geological location. Thus in the embedding space, the dimensions describing the "capital" attributes will be close, but those describing population and location attributes will be distant.

Therefore, the space of a domain reflected in the vector space can be an enclosure with narrow boundaries in some dimensions and with medium or wide boundaries in others. Figure 1 illustrates the observation. This fits the shape of a hyper-ellipsoid, which can rotate freely.

4.2   Ellipsoid in Hyper-Dimensional Space

A hyper-ellipsoid can be represented as:

$$(x - a)^T M(x - a) = 1 \tag{6}$$

where $x, a \in R^n$, and $M \in R^{n*n}$. M should be a positive definite matrix. According to Cholesky decomposition,

$$M = LL^T \tag{7}$$

where L should be a lower triangular matrix. In the training period, we decompose M to L, and update L instead of M. L will still be a lower triangular matrix in training. Using L to calculate M, we can assure that M remains a Hermitian matrix in the training period.

4.3   Distance Between Entity Vectors and Hyper-Ellipsoid

We use an approximate method to calculate the distance between a point and the surface of an ellipsoid in the vector space. First, a straight line is used to join the entity point with the geometric center of the hyper-ellipsoid. Then we work out the crossing point of the straight line and the hyper-ellipsoid's surface. Finally, the distance between the crossing point and the entity point is calculated, which is our defined distance between the entity vector and the hyper-ellipsoid. Figure 2 illustrates the distance.
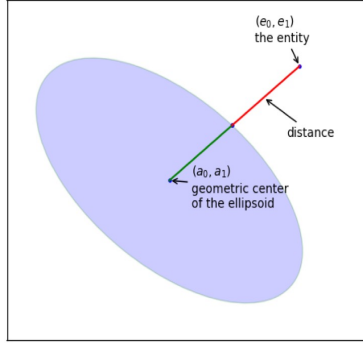
Fig. 2: An entity point and the surface of an ellipsoid, the red line is our defined distance.

The distance between the entity vector and the surface of the ellipsoid is:

$$D = \left| 1 - \frac{1}{\sqrt{(e-a)^T M (e-a)}} \right| \|e - a\|_2 \tag{8}$$

where $e, a \in R^n$, and $M \in R^{n*n}$, e is a vector representing the entity, and a is a vector representation of the hyper-ellipsoid's geometric centre.

Note that the use an approximate distance is mainly because the exact distance is complex with many constraints, which can be infeasibly slow to calculate.

4.4   Score Functions

The fitness between an entity vector and a domain is measured according to the distance D in equation 8. The lower the score, the better the entity fits the domain. In the training process, to fit the hyper-ellipsoid to a domain, we set the score function as:

$$f_{train}(e, E) = D \tag{9}$$

where e indicates an entity, E indicates an Ellipsoid and D indicates the geometric distance discussed above. Note that all es have been readily trained an a knowledge graph, and they will not change in ellipsoid-training. Only Ms and as are trained.

In the testing process, if an entity is inside the hyper-ellipsoid, it belongs to the domain, and we set f to 0; if an entity is outside the hyper-ellipsoid, it still can belong to the domain. We use D to describe the relatedness between the entity and the domain, setting $f = D$.

Thus, the score function in testing is:

$$f_{test}(e, E) = \begin{cases} 0 & (e-a)^T M(e-a) < 1 \\ D & (e-a)^T M(e-a) \geq 1 \end{cases} \tag{10}$$

where $(e-a)^T M(e-a) < 1$ means the entity is inside the hyper-ellipsoid, $(e-a)^T M (e-a) \geq 1$ means the entity is outside the hyper-ellipsoid, and D indicates the geometric distance.
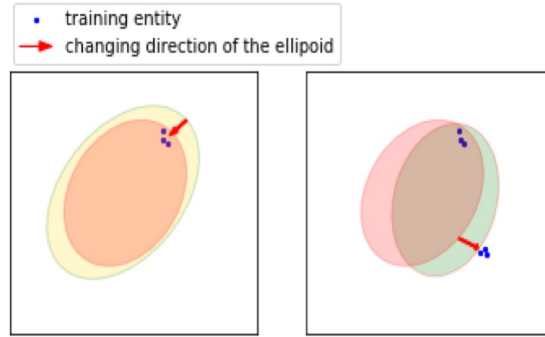
Fig. 3: Simulated training process. A hyper-ellipsoid changes its size, shape and orientation during training. The left figure shows that when training entities are inside the hyper-ellipsoid, boundaries of the ellipsoid will move inward. The right figure shows that when training entities are out of the hyper-ellipsoid, boundaries of the ellipsoid will move outward.

The training score ensures that the hyper-ellipsoid will fit the space of the domains and the testing score ensures that all entities belonging to a certain domain are treated equally, but the entities outside the domain are estimated by their distance from the domain.

4.5 Training

Our training goal is to find a hyper-ellipsoid that best fits a cluster of the entities that represents a domain. The training objective is:

$$L = \sum_{e \in Dom} \min \left(f_{\text{train}} \left(e, E \right) \right) \tag{11}$$

where e is an entity belonging to a domain Dom, and $min()$ aims to minimize the distance between the hyper-ellipsoid and the entities s. We use SGD to minimize f_{train} by updating the ellipsoid parameters.

For models having multiple spaces, we only choose the embeddings in the "final" space to train. For example, for TransR, we choose the projected space (relation space). So the model does not care about how different spaces are projected.

5   Combination with KG Embedding Models

Domain representations are combined with baseline knowledge graph embedding models to enhance the power of entity distinguish. When doing link prediction, entities not belonging to a domain receives a penalty over the baseline model scores based on their spatial distance from the domain. However, entities belonging to the domain do not receive such penalty scores.

In the testing process, we add the score function of DRE to the baseline models:

$$f_r \left(h, t \right) = f_{\text{test(baseline)}} \left(h, t \right) + f_{\text{test(DRE)}} \left(e, E \right) \tag{12}$$

where $f_{\text{test(baseline)}} \left(h, t \right)$ is the score function of the baseline model, and $f_{\text{test(DRE)}}(e, E)$ is the penalty score of our model.
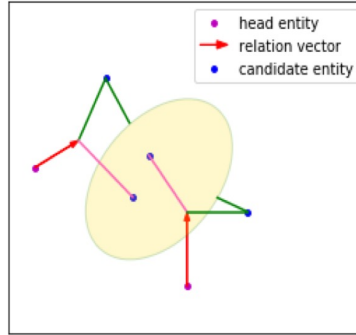
Fig. 4: The testing process of a translation-based model combined with DRE. For candidate entities inside the ellipsoidy, their scores have no change from the baseline models, which are indicated by the pink lines. However, for candidate entities outside the ellipsoid (outside the domain), their scores are the original scores augmented by the scores of DRE (the distance between the candidate entities to the ellipsoid's surface), which are the two intersecting green lines.

5.1   Example

We use STransE as an example to illustrate how our model DRE is applied on top of a knowledge graph embedding model. STransE is one of the best performing translation-based models, and its score function is $f_r(h, t) = \|W_{r,1}h + r - W_{r,2}t\|_{l_{1/2}}$, where $W_{r,1}$, $W_{r,2}t$ are the projection matrices.

The process consists of five steps:

1. Train a TransE model to obtain entity embeddings hs, ts and relation embeddings rs, mentioned in Equation 1.
2. Use the TransE embeddings to train STransE and obtain new entity and relation embeddings as well as the projection matrices, shown in Equation 3.
3. Use the entity embeddings and matrices from STransE (in the projected space) to train hyper-ellipsoids. Given a triple$< h, r, t >$, The projected embeddings of the two entities are used to train two domains - the head doamin and tail domain of the relation r.
4. For each testing case, calculate $f_{\text{test(STransE)}}$ and $f_{\text{test(DRE)}}$ respectively and add them for a final score $f = f_{\text{test(STransE)}} + f_{\text{test(DRE)}}$.
5. Use f in link prediction.

6   Experiments

We for comparing our domain representation model with the baseline models on link prediction, evaluating the results using mean rank, Hits@10/3/1 of TransE/TransR/STransE combined with DRE, respectively.

6.1   Datasets

We conduct our experiments on two typical knowledge graphs, namely WordNet [8] and Freebase [1], choosing the dataset WN18 from Wordnet and the dataset FB15K from Freebase. [1] Information of these two databases is given in Table 1.

Because of reverse relations, FB15K-237 and WN18RR have attracted much attention. However, DRE is free from this problem because each domain is independent. Besides, more papers

Dataset	#Ent	#Rel	#Train	#Valid	# Test	# Domain
WN18	40,943	18	141,442	5,000	5,000	36
FB15K	14,951	1,345	483,142	50,000	59,071	2690

Table 1: #Ent is the entity number of the database, #Rel is the relation number of the database. #Train/#Valid/#Test are the triples in training/validation/test sets.

Metric	external Information	WN18		FB15K	
		Mean Rank Filtered	Hits@10 Filtered	Mean Rank Filtered	Hits@10 Filtered
TransE [2]		251	89.2	125	47.1
TransH [14]		388	82.3	87	64.4
TransR [7]		225	92.0	77	68.7
TranSparse [5]		221	93.9	82	79.9
STransE [9]		206	93.4	69	79.7
TransA [6]	NA	**153**	-	58	-
ITransF [16]		205	95.2	65	81.4
DRE(TransE)		201	93.3	60	74.5
DRE(TransR)		165	96.5	38	82.1
DRE(STransE)		154	**96.9**	**36**	**83.4**

Table 2: Link prediction results on two datasets.

6.2   Evaluation

Based on link prediction, two measures are used as our evaluation metric. **Mean Rank**: the mean rank of correct entities in all test triples. **Hits@10/3/1**: the proportion of correct entities ranked at top 10/3/1 among all the candidate entities.

Following [2] and [7], we divide over evaluation into **Raw** and **Filtered**. Some triples obtained by randomly replacing entities in gold triples are also correct, such triples exist in the training, validation or test sets. If we do not remove the corrupted triples when calculating the rank of correct triples, the evaluation method is called **Raw**; if we remove them, the evaluation method called **Filtered**. f The relations can be divided into four categories according to their head and tail entity counts. For any relation, if there is only one head entity in the knowledge graph when given a tail entity, and vice versa, it is an **1-to-1** relation. If there is only one head entity when given a tail entity but many tail entities when given a head entity, it is an **1-to-N** relation; Reverse to **N-to-1** relation; If there many head entities when given a tail entity and vice versa, it is an **N-to-N** relation.

In both Raw and Filtered data, for any type of relations, lower Mean Rank and higher Hits@n indicate the better results and performance.

6.3   Hyper-parameters

When training the baseline models, we choose the best-selected parameters presented in their original papers. A Translation-based model must contains five parameters, namely the word vector size k, the learning rate λ, the margin γ, the batch size B and dissimilarity measure d.

---

reported results on WN18 and FB15K than those on FB15k-237 and WN18RR.So we choose the more general and widely-used datasets WN18 and FB15K.

Model	WN18				FB15K			
	MR	H@10	H@3	H@1	MR	H@10	H@3	H@1
TransE	251	89.2	83.2	34.2	125	47.1	49.4	20.9
DRE(TransE)	204	93.3	86.2	68.9	60	74.5	57.5	41.9
TransR	225	92.0	89.8	48.6	77	68.7	50.4	21.5
DRE(TransR)	164	96.5	94.4	71.1	38	82.1	70.6	55.5
STransE	206	93.4	90.2	70.0	69	79.7	56.8	26.5
DRE(STransE)	154	96.9	94.7	75.3	36	83.4	72.4	58.7

Table 3: Results on WN18 and FB15K.MR is mean rank, H@10/3/1 are Hits@10/3/1. In the results of TransE, TransR, and STransE, for MR and Hits@10, we use the results reported in the original papers; For Hits@3/1, we use our own results.

- For TransE, the configurations are: $k = 50$, $\lambda = 0.001$, $\gamma = 2$, B $= 120$, and $d = L_1$ on WN18; $k = 50$, $\lambda = 0.001$, $\gamma = 1$, $B = 120$ and $d = L_1$ on FB15K.
- For TransR, the configurations are: $k = 50$, $\lambda = 0.001$, $\gamma = 4$, $B = 1440$, and $d = L_1$ on WN18; $k = 50$, $\lambda = 0.001$, $\gamma = 1$, $B = 4800$ and $d = L_1$ on FB15K.
- For STransE, the configurations are: $k = 50$, $\lambda = 0.0005$, $\gamma = 5$, $B = 120$, and $d = L_1$ on WN18; $k = 100$, $\lambda = 0.0001$, $\gamma = 1$, $B = 120$ and $d = L_1$ on FB15K.

In our DRE model, we run SGD for 500 epochs to estimate the parameters of hyper-ellipsoids, with the learning rate $\lambda \in \{ 0.000001; 0.00001; 0.0001; 0.001; \}$ and the batch size $B = 120$. The vector size k is set the same as the baseline model's vector size. We finally chose $\lambda = 0.00001$ as our learning rate according to the results. Except for λ and B, our model does not need other hyper-parameters.

6.4   Results

Table 2 shows the link prediction results of previous work and our method by the Mean Rank and Hit@10. The first 12 rows are prior models without external information; the next three rows are our model DRE with three baseline models,namely DRE(TransE), DRE(TransR) and DRE(STransE), respectively. Our model is also external information free. Models using external information can achieve better results compared with external-information-free models, but this is not a direct comparison.

Among models without any external information, our model has achieved best results in both mean rank and Hits@10 on WN18 and FB15K, for example, our model improves Hits@10 of STransE from 93.4%/79.7% to 96.9%/83.4% and lower Mean Rank from 206/69 to 154/36 on WN18/FB15K, respectively.

Mean Rank. For Mean Rank, our model shows strong benefits, significantly reducing extreme cases in link prediction. Taking tail prediction by TransE as an example. In some cases (which are frequent in our test set), $h+r$ can be very distant from the t in the hyperspace, which makes the rank of correct triples very high. However, when $h + r$ is far away from the right entity t, our model gives a large penalty to the incorrect entities around $h + r$, while only giving a small or none penalty to the correct entities, which are not away from the domain. In this way, the extreme cases are reduced significantly.

6.5   Discussion

Table 3 shows that our model gives improvements on every evaluation metric over the baseline models. Also, the results of DRE-(TransE) are roughly similar to the STransE

Model	Predicting Head (Hits@10)				Predicting Tail (Hits@10)			
	1-to-1 1.4%	1-to-N 8.9%	N-to-1 14.7%	N-to-N 75.0%	1-to-1 1.4%	1-to-N 8.9%	N-to-1 14.7%	N-to-N 75.0%
TransE	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
DRE(TransE)	**79.6**	**85.1**	**47.1**	**75.6**	**79.8**	**52.8**	**84.1**	**78.0**
TransR	78.8	89.2	34.1	69.2	79.2	38.4	90.4	72.1
DRE(TransR)	**86.2**	87.8	**63.1**	**81.5**	**88.1**	**71.0**	89.0	**85.7**
STransE	82.8	94.2	50.4	80.1	82.4	56.9	93.4	83.1
DRE(STransE)	**87.1**	88.9	**69.2**	**82.2**	**89.1**	**74.9**	89.2	**86.5**

Table 4: Experimental results on FB15K by mapping properties of relations.

results. Similar to STransE to some extent, we model the head and tail entities for each relation, respectively. Our model represents its head domain and tail domain, while STransE creates a head projection matrix and a tail projection matrix. STransE/TransR describe domain information by using projection matrices. However, their domain information is implicit, and difficult to ally beyond their models. Besides, they aim to separate entities in the same domain from each other. In contrast, our model aims to extract common knowledge of one domain, using it to separate entities in the domain from entities out of the domain. Notably, our model also improves Hits@1 over the baseline models significantly, especially on FB15K.

Hits@10. In Table 4, we analyze why our model works on Hits@10. On dataset FB15K, the testing triples with "1-to-1" relation take up only 1.4% of the dataset while the ratio is 8.9% for "1-to-N" relation, 14.7% for "N-to-1" relation and 75.0% for "N-to-N" relation. As a result, instances on "to-N" are much more than those on "to-1". Our model significantly boosts the performance of predicting "to-N" cases. In predicting head entities for "N-to-1" relations and tail entities for "1-to-N" relations, where the results in baseline models are relatively lower. For "N-to-N" relations, our model also improves over the baseline significantly. Since "to-N" domains account for 86.8% (8.9%/2+14.7%/2+75.0%) of testing instances, the overall results were also greatly enhanced.

For some cases in predicting "to-1", our model gives lower accuracies. It may be because lack of training data for those domains. For example, for "1-to-N" relations, the head entities are much fewer than the tail entities, which leads to lack of entities in head domains. As a result, the domains may not be well trained. The same occurs to the tail domains of "N-to-1" relations. But for "to-N" domains, the training entities are much more, and the ellipsoids are better trained to represent the domains.

The baseline models are weak in predicting the head of "N-to-1" and predicting the tail of "1-to-N" relations, which is the major source of errors. Our method addresses such weakness.

7 Conclusion

We have shown that a conceptually simple domain model is effective for enhancing the embeddings of knowledge graph by offering hierarchical knowledge. In addition, hyper-ellipsoids are used to represent domains in the vector space, and the distance between an entity and certain domain is used to infer whether the entity belongs to the domain and how much discrepancy they have. Our model can be used over various other

KG embedding models to help improve their performance. Results on link prediction show that our model significantly improves the accuracies of state-of-the-art baseline knowledge graph embeddings.

To our knowledge, we are the first to learn explicit hierarchical knowledge structure over knowledge graph embeddings. Future work includes extending our model to other related NLP problems, such as information extraction.

References

[1] Bollacker, K., al, E.: Freebase:a collaboratively created graph database for structuring human knowledge. In: SIGMOD. pp. 1247–1250 (2008)

[2] Bordes, A.e.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26

[3] Huang, J., Zhao, W.X., Dou, H., Wen, J.R., Chang, E.Y.: Improving sequential recommendation with knowledge-enhanced memory networks. In: The 41st International ACM SIGIR Conference on Research

[4] Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: IJCNLP. pp. 687–696 (2015)

[5] Ji, G., Liu, K., He, S., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: Thirtieth AAAI Conference. pp. 985–991 (2016)

[6] Jia, Y., Wang, Y., Lin, H., Jin, X., Cheng, X.: Locally adaptive translation for knowledge graph embedding. In: 30th AAAI. pp. 992–998 (2016)

[7] Lin, Y., Liu, Z., Zhu, X., Zhu, X., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: 29th AAAI. pp. 2181–2187 (2015)

[8] Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM

[9] Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: STransE: a novel embedding model of entities and relationships in knowledge bases. In: HLT-NAACL (2016)

[10] Shen, Y., Deng, Y., Yang, M., Li, Y., Du, N., Fan, W., Lei, K.: Knowledge-aware attentive neural network for ranking question answer pairs. pp. 901–904 (06 2018)

[11] Shi, B., Weninger, T.: ProjE: Embedding projection for knowledge graph completion. In: AAAI. vol. 17, pp. 1236–1242 (2017)

[12] Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: NIPS 26

[13] Turney, P.D.: Domain and function: A dual-space model of semantic relations and compositions. Journal of Artificial Intelligence Research **44**, 533–585 (2012)

[14] Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: 28th AAAI. pp. 1112–1119 (2014)

[15] Xiao, H., Huang, M., Zhu, X.: TransG: A generative model for knowledge graph embedding. In: 54th ACL. vol. 1, pp. 2316–2325 (2016)

[16] Xie, Q., Ma, X., Dai, Z., Hovy, E.: An interpretable knowledge transfer model for knowledge base completion. In: 55th ACL. pp. 950–962 (2017)

[17] Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR 2015 (2015)

[18] Yang, L., Qiu, M., Qu, C., Guo, J., Zhang, Y., Croft, W.B., Huang, J., Chen, H.: Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In: SIGIR '18 (2018)