Differentiating Concepts and Instances for Knowledge Graph Embedding

Xin Lv, Lei Hou*, Juanzi Li, Zhiyuan Liu
Department of Computer Science and Technology,
Tsinghua University, China 100084
{lv-x18@mails.,houlei@,lijuanzi@,liuzy@}tsinghua.edu.cn

Abstract

Concepts, which represent a group of different instances sharing common properties, are essential information in knowledge representation. Most conventional knowledge embedding methods encode both entities (concepts and instances) and relations as vectors in a low dimensional semantic space equally, ignoring the difference between concepts and instances. In this paper, we propose a novel knowledge graph embedding model named TransC by differentiating concepts and instances. Specifically, TransC encodes each concept in knowledge graph as a sphere and each instance as a vector in the same semantic space. We use the relative positions to model the relations between concepts and instances (i.e., `instanceOf`), and the relations between concepts and sub-concepts (i.e., `subClassOf`). We evaluate our model on both link prediction and triple classification tasks on the dataset based on YAGO. Experimental results show that TransC outperforms state-of-the-art methods, and captures the semantic transitivity for `instanceOf` and `subClassOf` relation. Our codes and datasets can be obtained from `https://github.com/davidlvxin/TransC`.

1 Introduction

Knowledge graphs (KGs) aim at semantically representing the world's truth in the form of machine-readable graphs composed of triple facts. Knowledge graph embedding encodes each element (entities and relations) in knowledge graph into a continuous low-dimensional vector space. The learned representations make the knowledge graph essentially computable and have been proved to be helpful for knowledge graph completion and information extraction (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015, 2016).

*corresponding author

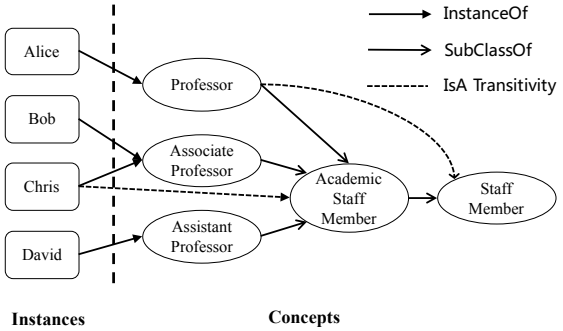


Figure 1: An example of concepts, instances, and isA transitivity.

In recent years, various knowledge graph embedding methods have been proposed, among which the translation-based models are simple and effective with good performances. Inspired by word2vec (Mikolov et al., 2013), given a triple (h, r, t), TransE learns vector embeddings $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$ which satisfy $\mathbf{r} \approx \mathbf{t} - \mathbf{h}$. Afterwards, TransH (Wang et al., 2014), TransR/CTransR (Lin et al., 2015b) and TransD (Ji et al., 2015), etc are proposed to address the problem of TransE when modeling 1-to-N, N-to-1, and N-to-N relations. As extensions of RESCAL(Nickel et al., 2011), which is a bilinear model, HolE(Nickel et al., 2016), DistMult(Yang et al., 2014) and ComplEx(Trouillon et al., 2016) achieve the state-of-the-art performances. Meanwhile, there are also some different methods using a variety of external information such as entity types (Xie et al., 2016), textual descriptions (Wang and Li, 2016), as well as logical rules to strengthen representations of knowledge graphs (Wang et al., 2015; Guo et al., 2016; Rocktäschel et al., 2015).

However, all these methods ignore to distinguish between concepts and instances, and regard both as entities to make a simplification. Actually, concepts and instances are organized differently in many real world datasets like YAGO (Suchanek

et al., 2007), Freebase (Bollacker et al., 2008), and WordNet (Miller, 1995). Hierarchical concepts in these knowledge bases provide a natural way to categorize and locate instances. Therefore, the common simplification in previous work will lead to the following two drawbacks:

Insufficient concept representation: Concepts are essential information in knowledge graph. A concept is a fundamental category of existence (Rosch, 1973) and can be reified by all of its actual or potential instances. Figure 1 presents an example of concepts and instances about university staffs. Most knowledge embedding methods encode both concepts and instances as vectors, cannot explicitly represent the difference between concepts and instances.

Lack transitivity of both isA relations: `instanceOf` and `subClassOf` (generally known as isA) are two special relations in knowledge graph. Different from most other relations, isA relations exhibit transitivity, e.g., the dotted lines in Figure 1 represent the facts inferred by isA transitivity. The indiscriminate vector representation for all relations in previous work cannot reserve this property well (see Section 5.3 for details).

To address these issues, we propose a novel translation embedding model named TransC in this paper. Inspired by (Tenenbaum et al., 2011), concepts in people's mind are organized hierarchically and instances should be close to concepts that they belong to. Hence in TransC, each concept is encoded as a sphere and each instance as a vector in the same semantic space, and relative positions are employed to model the relations between concepts and instances. More specifically, `instanceOf` relation is naturally represented by checking whether an instance vector is inside a concept sphere. For the `subClassOf` relation, we enumerate and quantify four possible relative positions between two concept spheres. We also define loss functions to measure the relative positions and optimize knowledge graph embeddings. Finally, we incorporate them into translation-based models to jointly learn the knowledge representations of concepts, instances and relations.

Experiments on real world datasets extracted from YAGO show that TransC outperforms previous work like TransE, TransD, HolE, DistMult and ComplEx in most cases. The contributions of this paper can be summarized as follows:

1. To the best of our knowledge, we are the first to propose and formalize the problem of knowledge graph embedding which differentiates between concepts and instances.

2. We propose a novel knowledge embedding method named TransC, which distinguishes between concepts and instances and deals with the transitivity of isA relations.

3. We construct a new dataset based on YAGO for evaluation. Experiments on link prediction and triple classification demonstrate that TransC successfully addresses the above problems and outperforms state-of-the-art methods.

2 Related Work

There are a variety of models for knowledge graph embedding. We divide them into three kinds and introduce them respectively.

2.1 Translation-based Models

TransE (Bordes et al., 2013) regards a relation $\mathbf{r}$ as a translation from $\mathbf{h}$ to $\mathbf{t}$ for a triple (h, r, t) in training set. The vector embeddings of this triple should satisfy $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Hence, $\mathbf{t}$ should be the nearest neighbor of $\mathbf{h} + \mathbf{r}$, and the loss function is

$$f_r(h, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2^2. \quad (1)$$

TransE is suitable for 1-to-1 relations, but it has problems when handling 1-to-N, N-to-1, and N-to-N relations.

TransH (Wang et al., 2014) attempts to alleviate the problems of TransE above. It regards a relation vector $\mathbf{r}$ as a translation on a hyperplane with $\mathbf{w}_r$ as the normal vector. The vector embeddings will be first projected to the hyperplane of relation r and get $\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r$ and $\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r$. The loss function of TransH is

$$f_r(h, t) = ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2. \quad (2)$$

TransR/CTransR (Lin et al., 2015b) addresses the issue in TransE and TransH that some entities are similar in the entity space but comparably different in other specific aspects. It sets a transfer matrix $\mathbf{M}_r$ for each relation r to map entity embedding to relation vector space. Its loss function is

$$f_r(h, t) = ||\mathbf{M}_r\mathbf{h} + \mathbf{r} - \mathbf{M}_r\mathbf{t}||_2^2. \quad (3)$$

TransD (Ji et al., 2015) considers the different types of entities and relations at the same time. Each relation-entity pair (r, e) will have a mapping matrix $\mathbf{M}_{re}$ to map entity embedding into relation vector space. And the projected vectors could be defined as $\mathbf{h}_\perp = \mathbf{M}_{rh}\mathbf{h}$ and $\mathbf{t}_\perp = \mathbf{M}_{rt}\mathbf{t}$. The loss function of TransD is

$$f_r(h,t) = ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2. \quad (4)$$

There are many other translation-based models in recent years. For example, TranSparse (Ji et al., 2016) simplifies TransR by enforcing the sparseness on the projection matrix, PTransE (Lin et al., 2015a) considers relation paths as translations between entities for representation learning, (Xiao et al., 2016a) proposes a manifold-based embedding principle (ManifoldE) for precise link prediction, TransF (Feng et al., 2016) regards relation as translation between head entity vector and tail entity vector with flexible magnitude, (Xiao et al., 2016b) proposes a new generative model TransG, and KG2E (He et al., 2015) uses Gaussian embedding to model the data uncertainty. All these models can be seen in (Wang et al.).

2.2 Bilinear Models

RESCAL(Nickel et al., 2011) is the first bilinear model. It associates each entity with a vector to capture its latent semantics. Each relation is represented as a matrix which models pairwise interactions between latent factors.

Many extensions of RESCAL have been proposed by restricting bilinear functions in recent years. For example, DistMult (Yang et al., 2014) simplifies RESCAL by restricting the matrices representing relations to diagonal matrices. HolE(Nickel et al., 2016) combines the expressive power of RESCAL with the efficiency and simplicity of DistMult. It represents both entities and relations as vectors in $\mathbb{R}^d$. ComplEx(Trouillon et al., 2016) extends DistMult by introducing complex-valued embeddings so as to better model asymmetric relations.

2.3 External Information Learning Models

External information like textual information is significant for knowledge representation. TEKE (Wang and Li, 2016) uses external context information in a text corpus to represent both entities and words into a joint vector space with alignment models. DKRL (Xie et al., 2016) directly learns entity representations from entity descriptions. (Wang et al., 2015; Guo et al., 2016; Rocktäschel et al., 2015) use logical rules to strengthen representations of knowledge graphs.

All models above do not differentiate between concepts and instances. To the best of our knowledge, our proposed TransC is the first attempt which represents concepts, instances, and relations differently in the same space.

3 Problem Formulation

In this section, we formulate the problem of knowledge graph embedding with concepts and instances. Before that, we first introduce the input knowledge graph.

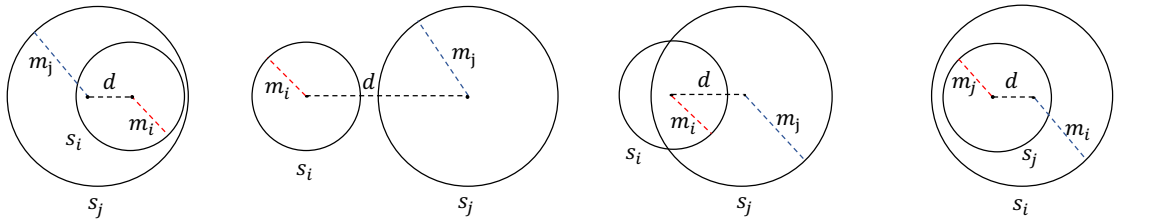
Knowledge Graph $\mathcal{KG}$ describes concepts, instances, and the relations between them. It can be formalized as $\mathcal{KG} = \{\mathcal{C}, \mathcal{I}, \mathcal{R}, \mathcal{S}\}$. $\mathcal{C}$ and $\mathcal{I}$ denote the sets of concepts and instances respectively. Relation set $\mathcal{R}$ can be formalized as $\mathcal{R} = \{r_e, r_c\} \cup \mathcal{R}_l$, where r_e is an `instanceOf` relation, r_c is a `subClassOf` relation, and $\mathcal{R}_l$ is the instance relation set. Therefore, the triple set $\mathcal{S}$ can be divided into three disjoint subsets:

1. `InstanceOf` triple set $\mathcal{S}_e = \{(i, r_e, c)_k\}_{k=1}^{n_e}$, where $i \in \mathcal{I}$ is an instance, $c \in \mathcal{C}$ is a concept, and n_e is the size of $\mathcal{S}_e$.

2. `SubClassOf` triple set $\mathcal{S}_c = \{(c_i, r_c, c_j)_k\}_{k=1}^{n_c}$, where $c_i, c_j \in \mathcal{C}$ are concepts, c_i is a sub-concept of c_j, and n_c is the size of $\mathcal{S}_c$.

3. Relational triple set $\mathcal{S}_l = \{(h, r, t)_k\}_{k=1}^{n_l}$, where $h, r \in \mathcal{I}$ are head instance and tail instance, $r \in \mathcal{R}_l$ is an instance relation, and n_l is the size of $\mathcal{S}_l$.

Given knowledge graph $\mathcal{KG}$, **knowledge graph embedding with concepts and instances** aims at learning embeddings for instances, concepts, and relations in the same space $\mathbb{R}^k$. For each concept $c \in \mathcal{C}$, we learn a sphere $s(\mathbf{p}, m)$ with $\mathbf{p} \in \mathbb{R}^k$ and m denoting the sphere center and radius. For each instance $i \in \mathcal{I}$ and instance relation $r \in \mathcal{R}_l$, we learn a low-dimensional vector $\mathbf{i} \in \mathbb{R}^k$ and $\mathbf{r} \in \mathbb{R}^k$ respectively. Specifically, the `instanceOf` and `subClassOf` representations are well-designed so that the transitivity of isA relations can be reserved, namely, `instanceOf-subClassOf`



(a) $d < |m_i - m_j| \wedge m_i < m_j$ (GOAL) (b) $d \geq |m_i + m_j|$ (c) $|m_i - m_j| \leq d < |m_i + m_j|$ (d) $d < |m_i - m_j| \wedge m_i \geq m_j$

Figure 2: Four relative positions between sphere s_i and s_j.

transitivity shown in Equation 5:

$$(i, r_e, c_1) \in S_e \wedge (c_1, r_c, c_2) \in S_c \rightarrow (i, r_e, c_2) \in S_e, \quad (5)$$

and `subClassOf`-`subClassOf` transitivity shown in Equation 6:

$$(c_1, r_c, c_2) \in S_c \wedge (c_2, r_c, c_3) \in S_c \rightarrow (c_1, r_c, c_3) \in S_c. \quad (6)$$

Based on the definition, how to model concepts and isA relations is critical to solve this problem.

4 Our Approach

To differentiate between concepts and instances for knowledge graph embedding, we propose a novel method named TransC. We define different loss functions to measure the relative positions in embedding space, and then jointly learn the representations of concepts, instances, and relations based on the translation-based models.

4.1 TransC

We have three kinds of triples in our triple set $\mathcal{S}$ and define different loss function for them respectively.

InstanceOf Triple Representation. For a given `instanceOf` triple (i, r_e, c), if it is a true triple, $\mathbf{i}$ should be inside the sphere s to represent the `instanceOf` relation between them. Actually, there is another relative position that $\mathbf{i}$ is outside the sphere s. In this condition, the embeddings still need to be optimized. The loss function is defined as

$$f_e(i, c) = ||\mathbf{i} - \mathbf{p}||_2 - m. \quad (7)$$

SubClassOf Triple Representation. For a `subClassOf` triple (c_i, r_c, c_j), just like before, concepts c_i, c_j are encoded as spheres $s_i(\mathbf{p}_i, m_i)$ and $s_j(\mathbf{p}_j, m_j)$. We first denote the distance between the centers of the two spheres as

$$d = ||\mathbf{p}_i - \mathbf{p}_j||_2. \quad (8)$$

If (c_i, r_c, c_j) is a true triple, sphere s_i should be inside sphere s_j (Figure 2a) to represent the `subClassOf` relation between them. Actually, there are three other relative positions between sphere s_i and s_j (as shown in Figure 2). We also have three loss functions under these three conditions:

1. s_i is separate from s_j (Figure 2b). The embeddings still need to be optimized. In this condition, the two spheres need to get closer in optimalization. Therefore, the loss function is defined as

$$f_c(c_i, c_j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 + m_i - m_j. \quad (9)$$

2. s_i intersects with s_j (Figure 2c). This condition is similar to condition 1. The loss function is defined as

$$f_c(c_i, c_j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 + m_i - m_j. \quad (10)$$

3. s_j is inside s_i (Figure 2d). It is different from our target and we should reduce m_j and increase m_i. Hence, the loss function is

$$f_c(c_i, c_j) = m_i - m_j. \quad (11)$$

Relational Triple Representation. For a relational triple (h, r, t), TransC will learn low-dimensional vectors $\mathbf{h}, \mathbf{t}, \mathbf{r} \in \mathbb{R}^k$ for instances and relations. Just like TransE (Bordes et al., 2013), the loss function of this kind of triples is defined as

$$f_r(h, t) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2^2. \quad (12)$$

After having embeddings above, TransC can easily deal with the transitivity of isA relations. If we have true triples (i, r_e, c_i) and (c_i, r_c, c_j), which means $\mathbf{i}$ is inside the sphere s_i and s_i is inside s_j, we can get a result that $\mathbf{i}$ is also inside the sphere s_j. It can be concluded that (i, r_e, c_j) is a

true triple and TransC can handle `instanceOf`-`subClassOf` transitivity. Similarly, if we have true triples (c_i, r_c, c_j) and (c_j, r_c, c_k), we can get a result that sphere s_i is inside sphere s_k. It means (c_i, r_e, c_k) is a true triple and TransC can deal with `subClassOf`-`subClassOf` transitivity.

In experiments, we enforce constrains as $||\mathbf{h}||_2 \leq 1$, $||\mathbf{r}||_2 \leq 1$, $||\mathbf{t}||_2 \leq 1$ and $||\mathbf{p}||_2 \leq 1$.

4.2 Training Method

For `instanceOf` triples, we use ξ and ξ' to denote a positive triple and a negative triple. $\mathcal{S}_e$ and $\mathcal{S}'_e$ are used to describe the positive triple set and negative triple set. Then we can define a margin-based ranking loss for `instanceOf` triples:

$$\mathcal{L}_e = \sum_{\xi \in \mathcal{S}_e} \sum_{\xi' \in \mathcal{S}'_e} [\gamma_e + f_e(\xi) - f_e(\xi')]_+, \quad (13)$$

where $[x]_+ \triangleq \max(0, x)$ and γ_e is the margin separating positive triplets and negative triplets. Similarly, for `subClassOf` triples, we will have a ranking loss:

$$\mathcal{L}_c = \sum_{\xi \in \mathcal{S}_c} \sum_{\xi' \in \mathcal{S}'_c} [\gamma_c + f_c(\xi) - f_c(\xi')]_+, \quad (14)$$

and for relational triples, we will have a ranking loss:

$$\mathcal{L}_l = \sum_{\xi \in \mathcal{S}_l} \sum_{\xi' \in \mathcal{S}'_l} [\gamma_l + f_r(\xi) - f_r(\xi')]_+. \quad (15)$$

Finally, we define the overall loss function as linear combinations of these three functions:

$$\mathcal{L} = \mathcal{L}_e + \mathcal{L}_c + \mathcal{L}_l. \quad (16)$$

The goal of training TransC is to minimize the above function, and iteratively update embeddings of concepts, instances, and concepts.

Every triple in our training set has a label to indicate whether the triple is positive or negative. But existing knowledge graph only contains positive triples. We need to generate negative triples by corrupting positive triples. For a relational triple (h, r, t), we replace h or t to generate a negative triple (h', r, t) or (h, r, t'). For example, we get h' by randomly picking from a set $\mathcal{M}_t = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_n$, where n is the number of concepts that t belongs to and $\mathcal{M}_i = \{a | a \in \mathcal{I} \wedge (a, r_e, c_i) \in \mathcal{S}_e \wedge (t, r_e, c_i) \in \mathcal{S}_e \wedge t \neq a\}$. For the other two kinds of triples, we follow the same policy to generate negative triples. We also use two strategies "unif" and "bern" described in (Wang et al., 2014) to replace instances or concepts.

DataSets	YAGO39K	M-YAGO39K
#Instance	39,374	39,374
#Concept	46,110	46,110
#Relation	39	39
#Relational Triple	354,997	354,997
#`InstanceOf` Triple	442,836	442,836
#`SubClassOf` Triple	30,181	30,181
#Valid (Relational Triple)	9,341	9,341
#Test (Relational Triple)	9,364	9,364
#Valid (`InstanceOf` Triple)	5,000	8,650
#Test (`InstanceOf` Triple)	5,000	8,650
#Valid (`SubClassOf` Triple)	1,000	1,187
#Test (`SubClassOf` Triple)	1,000	1,187

Table 1: Statistics of YAGO39K and M-YAGO39K.

5 Experiments and Analysis

We evaluate our method on two typical tasks commonly used in knowledge graph embedding: link prediction (Bordes et al., 2013) and triple classification (Socher et al., 2013).

5.1 Datasets

Most previous work used FB15K and WN18 (Bordes et al., 2013) for evaluation. But these two datasets are not suitable for our model because FB15K mainly consists of instances and WN18 mainly contains concepts. Therefore, we use another popular knowledge graph YAGO (Suchanek et al., 2007) for evaluation, which contains a lot of concepts from WordNet and instances from Wikipedia. We construct a subset of YAGO named YAGO39K for evaluation through the following steps:

(1) We randomly select some relational triples like (h, r, t) from the whole YAGO dataset as our relational triple set $\mathcal{S}_l$.

(2) For every instance and instance relation existed in our relational triples, we save it to construct instance set $\mathcal{I}$ and instance relation set $\mathcal{R}_l$ respectively.

(3) For every `instanceOf` triple (i, r_e, c) in YAGO, if $i \in \mathcal{I}$, we save this triple to construct `instanceOf` triple set $\mathcal{S}_e$.

(4) For every concept existed in `instanceOf` triple set $\mathcal{S}_e$, we save it to construct concept set $\mathcal{C}$.

(5) For every `subClassOf` triple (c_i, r_c, c_j) in YAGO, if $c_i \in \mathcal{C} \wedge c_j \in \mathcal{C}$, we save this triple to construct `subClassOf` triple set $\mathcal{S}_c$.

(6) Finally, we achieve our triple set $\mathcal{S} = \mathcal{S}_e \cup \mathcal{S}_c \cup \mathcal{S}_l$ and our relation set $\mathcal{R} = \{r_e, r_c\} \cup \mathcal{R}_l$.

To evaluate every model's performance in handling the transitivity of isA relations, we generate some new triples based on YAGO39K using the transitivity of isA relations. These new triples will

Experiments	Link Prediction					Triple Classification(%)			
Metric	MRR		Hits@N(%)			Accuracy	Precision	Recall	F1-Score
	Raw	Filter	1	3	10				
TransE	0.114	0.248	12.3	28.7	51.1	92.1	92.8	91.2	92.0
TransH	0.102	0.215	10.4	24.0	45.1	90.8	91.2	90.3	90.8
TransR	0.112	0.289	15.8	33.8	56.7	91.7	91.6	91.9	91.7
TransD	0.113	0.176	8.9	19.0	35.4	89.3	88.1	91.0	89.5
HolE	0.063	0.198	11.0	23.0	38.4	92.3	92.6	91.9	92.3
DistMult	**0.156**	0.362	22.1	43.6	66.0	93.5	93.9	93.0	93.5
ComplEx	0.058	0.362	29.2	40.7	48.1	92.8	92.6	**93.1**	92.9
TransC (unif)	0.087	**0.421**	28.3	50.0	69.2	93.5	94.3	92.6	93.4
TransC (bern)	0.112	0.420	**29.8**	**50.2**	**69.8**	**93.8**	**94.8**	92.7	**93.7**

Table 2: Experimental results on link prediction and triple classification for relational triples. Hits@N uses results of "Filter" evaluation setting.

be added to valid and test datasets of YAGO39K to create a new dataset named M-YAGO39K. Specific steps are described as follows:

(1) For every `instanceOf` triple (i, r_e, c) in valid and test dataset, if (c, r_c, c_j) exists in training dataset, we save a new `instanceOf` triple (i, r_e, c_j).

(2) For every `subClassOf` triple (c_i, r_c, c_j) in valid and test dataset, if (c_j, r_c, c_k) exists in training dataset, we save a new `subClassOf` triple (c_i, r_c, c_k).

(3) We add these new triples to valid and test dataset of YAGO39K to get M-YAGO39K.

The statistics of YAGO39K and M-YAGO39K are shown in Table 1.

5.2 Link Prediction

Link Prediction aims to predict the missing h or t for a relational triple (h, r, t). In this task, we need to give a ranking list of candidate instances from the knowledge graph, instead of only giving one best result.

For every test relational triple (h, r, t), we remove the head or tail instance and replace it with all instances existed in knowledge graph, and rank these instances in ascending order of distances calculated by loss function f_r. Just like (Bordes et al., 2013), we use two evaluation metrics in this task: (1) the mean reciprocal rank of all correct instances (MRR) and (2) the proportion of correct instances that rank no larger than N (Hits@N). A good embedding model should achieve a high MRR and a high Hits@N. We note that a corrupted triple may also exist in knowledge graph, which should also be regarded as a correct prediction. However, the above evaluations do not handle this issue and may underestimate the results. Hence, we filter out every triple appeared in our knowledge graph before getting the ranking list. The first evaluation setting is called "Raw" and the second one is called "Filter." We report the experiment results on both settings.

In this task, we use dataset YAGO39K for evaluation. We select learning rate λ for SGD among $\{0.1, 0.01, 0.001\}$, the three margins γ_l, γ_e and γ_c among $\{0.1, 0.3, 0.5, 1, 2\}$, the dimension of instance vectors and relation vectors n among $\{20, 50, 100\}$. The best configurations are determined according to the Hits@10 in valid set. The optimal configurations are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 1$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. We train every model for 1000 rounds in this task.

Evaluation results on YAGO39K are shown in Table 2. From the table, we can conclude that: (1) TransC significantly outperforms other models in terms of Hits@N. This indicates that TransC can use isA triples' information better than other models, which is helpful for instance representation learning. (2) TransC performs a little bit worse than DistMult in some settings. The reason may be that we determine the best configurations only according to the Hits@10, which may lead to a low MRR. (3) The "bern" sampling trick works well for TransC.

5.3 Triple Classification

Triple Classification aims to judge whether a given triple is correct or not, which is a binary classification task. This triple can be a relational triple, an `instanceOf` triple or a `subClassOf` triple.

Negative triples are needed for evaluation of binary classification. Hence, we construct some negative triples following the same setting in (Socher et al., 2013). There are as many true triples as negative triples in both valid and test set.

For triple classification, we set a threshold δ_r for every relation r. For a given test triple, if its

Datasets	YAGO39K				M-YAGO39K			
Metric	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
TransE	82.6	83.6	81.0	82.3	71.0↓	81.4↓	54.4↓	65.2↓
TransH	82.9	83.7	81.7	82.7	70.1↓	80.4↓	53.2↓	64.0↓
TransR	80.6	79.4	**82.5**	80.9	70.9↓	73.0↓	66.3↓	69.5↓
TransD	83.2	84.4	81.5	82.9	72.5↓	73.1↓	71.4↓	72.2↓
HolE	82.3	86.3	76.7	81.2	74.2↓	81.4↓	62.7↓	70.9↓
DistMult	**83.9**	**86.8**	80.1	**83.3**	70.5↓	86.1↓	49.0↓	62.4↓
ComplEx	83.3	84.8	81.1	82.9	70.2↓	84.4↓	49.5↓	62.4↓
TransC (unif)	80.2	81.6	80.0	79.7	**85.5**↑	**88.3**↑	81.8↑	85.0↑
TransC (bern)	79.7	83.2	74.4	78.6	85.3↑	86.1↑	**84.2**↑	**85.2**↑

Table 3: Experimental results on `instanceOf` triple classification(%).

Datasets	YAGO39K				M-YAGO39K			
Metric	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score
TransE	77.6	72.2	89.8	80.0	76.9↓	72.3↑	87.2↓	79.0↓
TransH	80.2	76.4	87.5	81.5	79.1↓	72.8↓	92.9↑	81.6↑
TransR	80.4	74.7	91.9	82.4	80.0↓	73.9↓	92.9↑	82.3↓
TransD	75.9	70.6	88.8	78.7	76.1↑	70.7↑	89.0↑	78.8↑
HolE	70.5	73.9	63.3	68.2	66.6↓	72.3↓	53.7↓	61.7↓
DistMult	61.9	68.7	43.7	53.4	60.7↓	71.7↑	35.5↓	47.7↓
ComplEx	61.6	71.5	38.6	50.1	59.8↓	65.6↓	41.4↑	50.7↑
TransC (unif)	82.9	77.1	93.7	84.6	83.0↑	77.5↑	**93.1**↓	84.7↑
TransC (bern)	**83.7**	**78.1**	**93.9**	**85.2**	**84.4**↑	**80.7**↑	90.4↓	**85.3**↑

Table 4: Experimental results on `subClassOf` triple classification(%).

loss function is smaller than δ_r, it will be classified as positive, otherwise negative. δ_r is obtained by maximizing the classification accuracy on valid set.

In this task, we use dataset YAGO39K and M-YAGO39K for evaluation. Parameters are selected in the same way as in link prediction. The best configurations are determined by accuracy in valid set. The optimal configurations for YAGO39K are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 0.1$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. The optimal configurations for M-YAGO39K are: $\gamma_l = 1$, $\gamma_e = 0.1$, $\gamma_c = 0.3$, $\lambda = 0.001$, $n = 100$ and taking L_2 as dissimilarity. For both datasets, we traverse all the training triples for 1000 rounds.

Our datasets have three kinds of triples. Hence, we do experiments on them respectively. Experimental results for relational triples, `instanceOf` triples, and `subClassOf` triples are shown in Table 2, Table 3, and Table 4 respectively. In Table 3 and Table 4, a rising arrow means performance of this model have a promotion from YAGO39K to M-YAGO39K and a down arrow means a drop.

From Table 2, we can learn that: (1) TransC outperforms all previous work in relational triple classification. (2) The "bern" sampling trick works better than "unif" in TransC.

From Table 3 and Table 4, we can conclude that: (1) On YAGO39K, some compared models perform better than TransC in `instanceOf` triple classification. This is because that `instanceOf` has most triples (53.5%) among all relations in YAGO39K. This relation is trained superabundant times and nearly achieves the best performance, which has an adverse effect on other triples. TransC can find a balance between them and all triples achieve a good performance. (2) On YAGO39K, TransC outperforms other models in `subClassOf` triple classification. As shown in Table 1, `subClassOf` triples are much less than `instanceOf` triples. Hence, other models can not achieve the best performance under the bad influence of `instanceOf` triples. (3) On M-YAGO39K, TransC outperforms previous work in both `instanceOf` triple classification and `subClassOf` triple classification, which indicates that TransC can handle the transitivity of isA relations much better than other models. (4) After comparing experimental results in YAGO39K and M-YAGO39K, we can find that most previous work's performance suffers a big drop in `instanceOf` triple classification and a small drop in `subClassOf` triple classification. This shows that previous work can not deal with `instanceOf-subClassOf` transitivity well. (5) In TransC, nearly all performances have a significant promotion from YAGO39K to M-YAGO39K. Both `instanceOf-subClassOf` transitivity and `subClassOf-subClassOf` transitivity are solved well in TransC.

5.4 Case Study

We have shown that TransC have a good performance for knowledge graph embedding and dealing with transitivity of isA relations. In this section, we present an example of finding new `instanceOf` triples and `subClassOf` triples using results of TransC.

As shown in Figure 3, *New York City* is an instance and others are concepts. The solid lines represent the triples from our datasets and the dotted lines represent the facts inferred by our model. TransC can find two new `instanceOf` triples (*New York City*, `instanceOf`, *City*) and (*New York City*, `instanceOf`, *Municipality*). It can also find a new `subClassOf` triple (*Port Cities*, `subClassOf`, *City*). Following the transitivity of isA relations, we can know all these three new triples are right. Unfortunately, most previous work regards these three triples as wrong, which means they can not handle transitivity of isA relations well.

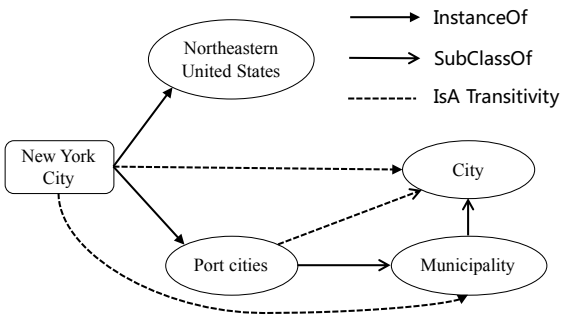


Figure 3: An inference example of TransC.

6 Conclusion and Future Work

In this paper, we propose a new knowledge embedding model named TransC. TransC embeds instances, concepts, and relations in the same space to deal with the transitivity of isA relations. We create a new dataset YAGO39K for evaluation. Experiment results show that TransC outperforms previous translation-based models in most cases. Besides, It can also handle the transitivity of isA relations much better than other models. In our future work, we will explore the following research directions: (1) Sphere is a simple model to represent a concept in semantic space, but it still have some limits since it is too naive. we will try to find a more expressive model instead of spheres to represent concepts. (2) A concept may have different meanings in different triples. We will try to use several typical vectors of instances as a concept's centers to represent different meanings of a concept. Then a concept can have different embeddings in different triples.

Acknowledgments

The work is supported by NSFC key project (No. 61533018, U1736204, 61661146007), Ministry of Education and China Mobile Research Fund (No. 20181770250), and THUNUS NExT Co-Lab.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. 2016. Knowledge graph embedding by flexible translation. In *KR*.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *EMNLP*.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *CIKM*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL*.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *AAAI*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*.

Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting logical background knowledge into embeddings for relation extraction. In *NAACL*.

Eleanor H. Rosch. 1973. Natural categories. *Cognitive Psychology*, 4(3):328–350.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. 2011. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *IJCAI*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

Zhigang Wang and Juan-Zi Li. 2016. Text-enhanced representation learning for knowledge graph. In *IJCAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016a. From one point to a manifold: knowledge graph embedding for precise link prediction. In *AAAI*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016b. Transg: A generative model for knowledge graph embedding. In *ACL*.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.