TransFG: A Fine-Grained Model for Knowledge Graph Embedding

Yaowei Yu, Zhuoming Xu(✉), Yan Lv, and Jian Li

College of Computer and Information, Hohai University, Nanjing 210098, China
{ywyu, zmxu, ylv, jli}@hhu.edu.cn

Abstract. Although concepts and instances in a knowledge graph (KG) are distinguished, TransC embeds concepts, instances, and various relations into the same vector space, which leads to the following problems: (1) The same instance in different triples that model different relations between instances is represented as the same vector, resulting in improper representation of different properties possessed by this instance; (2) Multiple instances not belonging to one concept may be located in the sphere representing this concept, resulting in an inaccurate modeling of the instanceOf relations between these instances and the concept. Based on TransC, this paper proposes a fine-grained KG embedding model called TransFG. TransFG embeds concepts, instances, and relations into different vector spaces and projects the instance vectors from the instance space to the concept space and the relation spaces through dynamic mapping matrices. This causes the projected vectors of the same instance in different triples to have different representations and the projected vectors of multiple instances belonging to the same concept to be spatially close to each other; otherwise they are far away. Experiments on the YAGO39K and M-YAGO39K datasets show that on the triple classification task, TransFG outperforms TransC and other typical KG embedding models in terms of accuracy, precision, recall and F1-score in most cases, and on the link prediction task, TransFG outperforms these compared models in terms of MRR and Hits@N in most cases.

Keywords: Knowledge graph embedding · Vector space · Mapping matrix · Triple classification · Link prediction

1 Introduction

A knowledge graph (KG) [1, 2] is a multi-relational graph consisting of entities (represented as nodes) and relations (represented as edges) between entities. The facts in KG are normally expressed as RDF triples. KGs like YAGO [3] and DBpedia [4, 5] have been widely used in knowledge-based applications such as question answering.

KG embedding [1, 2] is aimed at embedding entities and relations into continuous, low-dimensional vector space for efficiently performing downstream tasks such as link prediction and triple classification. According to the different types of scoring functions, there are two categories of KG embedding models: translational distance models and semantic matching models [1, 2]. TransE [6] is a representative of translation distance models. TransE has flaws in dealing with multi-mapping relations (one-to-many,

© Springer Nature Switzerland AG 2019

https://doi.org/10.1007/978-3-030-30952-7_45

many-to-one, and many-to-many). Based on TransE, many improved models have been proposed, for example, TransH [7], TransR [8], and TransD [9]. HolE [10], DistMult [11], and ComplEx [12] are typical semantic matching models.

Lv et al. [13] pointed out that existing KG embedding models fail to distinguish between concepts and instances, leading to some problems. Hence, they proposed a new model called TransC, which distinguishes between concepts and instances, and divides the triples in a KG into three disjoint subsets: the `instanceOf` triple set, the `subClassOf` triple set, and inter-instance relation triple set. However, TransC embeds concepts, instances, and various relations into the same vector space, which leads to the following problems: (1) The same instance in different triples that model different inter-instance relations is represented as the same vector, resulting in improper representation of different properties possessed by this instance; (2) Multiple instances not belonging to one concept may be located in the sphere representing this concept, resulting in an inaccurate modeling of the `instanceOf` relations between these instances and the concept.

Based on TransC, this paper proposes a fine-grained KG embedding model called TransFG. TransFG embeds concepts, instances, and relations into different vector spaces and projects the instance vectors from the instance space to the concept space and the relation spaces through dynamic mapping matrices. This causes the projected vectors of the same instance in different triples to have different representations and the projected vectors of multiple instances belonging to the same concept to be spatially close to each other; otherwise they are far away. We used two typical KG downstream tasks, triple classification and link prediction, to compare and evaluate TransFG, TransC, and several KG embedding models including TransE, TransH, TransR, TransD, HolE, DistMult, and ComplEx on the YAGO39K and M-YAGO39K datasets [13]. The experimental results show that on the triple classification task, TransFG outperforms TransC and other typical embedding models in terms of accuracy, precision, recall and F1-score in most cases, and on the link prediction task, TransFG outperforms these compared models in terms of MRR (the mean reciprocal rank of all correct instances) and Hits@N (the proportion of correct instances in the top-N ranked instances) in most cases.

2 TransFG: A Fine-Grained Model

In this section, we expatiate on our proposed TransFG model. We first briefly explain the basic idea of the model, then describe the model in detail, and finally explain the method of model training.

2.1 Basic Idea of TransFG

Like TransC, TransFG divides KG triples into three types: the `instanceOf` triples, the `subClassOf` triples, and inter-instance relation triples, and defines different loss functions for each type of triples. The main difference between TransFG and TransC is the improvement of the representations of the `instanceOf` triples and inter-instance relation triples. This is mainly achieved by embedding concepts, instances, and various relations between them into different spaces and applying corresponding mapping matrices.

For the representation of the `instanceOf` triples, TransFG projects instance vectors from the instance space to the concept space through dynamic mapping matrices. Let us use Fig. 1 to explain the representation of this type of triples. The meanings of the mathematical symbols in the figure are listed in Table 1. As shown in the figure, triangles, such as **e** and **f**, and pentagrams, such as **b**, denote different instance vectors belonging to two different concepts c_i and c_j, and $s_i(\mathbf{p}_i, m_i)$ and $s_j(\mathbf{p}_i, m_j)$ are two different concept spheres, respectively. For three `instanceOf` triples (e, r_e, c_i), (f, r_e, c_i), and (b, r_e, c_j), TransFG projects **e**, **f**, and **b** from the instance space to the concept space through the mapping matrices $\mathbf{M}_{ei}$, $\mathbf{M}_{fi}$, and $\mathbf{M}_{bj}$ and obtains the projected vectors $\mathbf{e}_\perp$, $\mathbf{f}_\perp$, and $\mathbf{b}_\perp$. If the three triples are positive triples (i.e., they exist in the KG), $\mathbf{e}_\perp$ and $\mathbf{f}_\perp$ are located in the sphere $s_i(\mathbf{p}_i, m_i)$, and $\mathbf{b}_\perp$ is located in sphere $s_j(\mathbf{p}_j, m_j)$. The loss functions for the `instanceOf` triples are then defined using the relative positions between the projected vectors and the concept spheres.

For the representation of the `subClassOf` triples, TransFG directly uses the corresponding method in TransC [13], that is, the loss functions for the `subClassOf` triples are defined using the relative positions between the two concept spheres.

For the representation of inter-instance relation triples, just like TransD [9], TransFG projects instance vectors from the instance space to the relation spaces through the corresponding mapping matrices. The loss functions for inter-instance relation triples are then defined using the projected vectors.

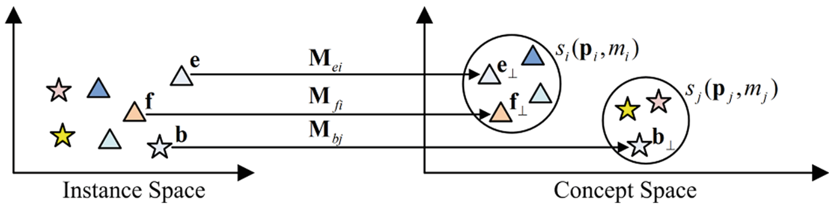


Fig. 1. Representation of the `instanceOf` triples in TransFG.

2.2 The TransFG Model

In this subsection, we describe the TransFG model in detail. We first list the mathematical symbols used to describe the model, and then define the loss functions for the three different types of triples: the `instanceOf` triples, the `subClassOf` triples, and inter-instance relation triples.

The mathematical symbols introduced in TransC's paper [13] and in our paper are listed in Table 1, where the symbols in the first eleven rows are introduced in [13].

Table 1. Mathematical symbols introduced in [13] and in our paper.

Symbols	Meanings
$\mathcal{KG} = \{\mathcal{C}, \mathcal{I}, \mathcal{R}, \mathcal{S}\}$	$\mathcal{KG}$ is a knowledge graph, where $\mathcal{C}$ denotes the concept set, $\mathcal{I}$ the instance set, $\mathcal{R}$ the relation set, and $\mathcal{S} = \mathcal{S}_e \cup \mathcal{S}_c \cup \mathcal{S}_l$ the set of triples existing in $\mathcal{KG}$ (i.e., the set of *positive triples*)
$\mathcal{R} = \{r_e, r_c\} \cup \mathcal{R}_l$	$\mathcal{R}$ is a relation set, where r_e is the `instanceOf` relation, r_c the `subClassOf` relation, and $\mathcal{R}_l$ the set of other relations
$\mathcal{S}_e = \{(i, r_e, c)_k\}_{k=1}^{n_e}$	$\mathcal{S}_e$ is the set of the positive `instanceOf` triples, where $i \in \mathcal{I}$ is an instance and $c \in \mathcal{C}$ a concept
$\mathcal{S}_c = \{(c_i, r_c, c_j)_k\}_{k=1}^{n_c}$	$\mathcal{S}_c$ is the set of the positive `subClassOf` triples, where both c_i and c_j are concepts
$\mathcal{S}_l = \{(h, r, t)_k\}_{k=1}^{n_l}$	$\mathcal{S}_l$ is the set of the positive inter-instance relation triples, where $h, t \in \mathcal{I}$ are the head instance and tail instance of a triple, respectively, and $r \in \mathcal{R}_l$ is an inter-instance relation
$\mathcal{S}' = \mathcal{S}'_e \cup \mathcal{S}'_c \cup \mathcal{S}'_l$	$\mathcal{S}'$ is the set of all negative triples, where $\mathcal{S}'_e, \mathcal{S}'_c, \mathcal{S}'_l$ are the sets of negative triples corresponding to $\mathcal{S}_e, \mathcal{S}_c, \mathcal{S}_l$, respectively
ξ, ξ'	$\xi \in \mathcal{S}$ is a positive triple, and $\xi' \in \mathcal{S}'$ is a negative triple
$\mathbf{i}$	$\mathbf{i} \in \mathbb{R}^n$ is the vector of the instance i in the triple (i, r_e, c)
$\mathbf{h}, \mathbf{r}, \mathbf{t}$	$\mathbf{h}, \mathbf{r}, \mathbf{t}$ are the vectors of the head instance h, relation r, and tail instance t in the inter-instance relation triple (h, r, t)
$s_i(\mathbf{p}_i, m_i)$	Sphere $s_i(\mathbf{p}_i, m_i)$ denote concept $c_i \in \mathcal{C}$, where $\mathbf{p}_i \in \mathbb{R}^k$ and m_i are the center and radius of the concept sphere, respectively
$\gamma_e, \gamma_c, \gamma_l$	$\gamma_e, \gamma_c, \gamma_l$ are the margins separating the positive triples and the negative triples in the three types of triples
k, n, z	k, n, z denote the dimensions of the vectors in the concept space, instance space and relation spaces, respectively
$\mathbf{i}_p, \mathbf{p}_p$	For a triple (i, r_e, c), $\mathbf{i}_p \in \mathbb{R}^n$ is the projection vector for the instance i and $\mathbf{p}_p \in \mathbb{R}^k$ the projection vector for the concept c
$\mathbf{h}_p, \mathbf{r}_p, \mathbf{t}_p$	For a triple (h, r, t), $\mathbf{h}_p, \mathbf{r}_p, \mathbf{t}_p$ are the projection vectors for the head instance h, relation r, and tail instance t, respectively
$\mathbf{M}_{ic}$	$\mathrm{M}_{ic} \in \mathbb{R}^{k \times n}$ is the mapping matrix that projects instance vector $\mathbf{i}$ to the concept space
$\mathbf{M}_{rh}, \mathbf{M}_{rt}$	$\mathbf{M}_{rh}, \mathbf{M}_{rt} \in \mathbb{R}^{z \times n}$ are the mapping matrices that projects the head instance and tail instance vectors $\mathbf{h}$ and $\mathbf{t}$ to the relation spaces
$\mathbf{i}_\perp$	$\mathbf{i}_\perp \in \mathbb{R}^k$ is the projected vector obtained by projecting the instance vector $\mathbf{i}$ to the concept space
$\mathbf{h}_\perp, \mathbf{t}_\perp$	$\mathbf{h}_\perp \in \mathbb{R}^z$, $\mathbf{t}_\perp \in \mathbb{R}^z$ are the projected vectors obtained by projecting the head instance vector $\mathbf{h}$ and tail instance vector $\mathbf{t}$ to the relation spaces, respectively

InstanceOf Triple Representation. For an `instanceOf` triple (i, r_e, c), TransFG learns the instance vector $\mathbf{i}$ of instance i, the projection vector $\mathbf{i}_p$ for i, the center vector $\mathbf{p}$ of the sphere representing concept c, and the projection vector $\mathbf{p}_p$ for c. The vectors $\mathbf{i}_p$ and $\mathbf{p}_p$ are used to construct the mapping matrix $\mathbf{M}_{ic} \in \mathbb{R}^{k \times n}$. TransFG projects $\mathbf{i}$ from the instance space to the concept space through the mapping matrix, which is defined as Eq. (1).

$$\mathbf{M}_{ic} = \mathbf{p}_p \mathbf{i}_p^\top + \mathbf{E}^{k \times n} \tag{1}$$

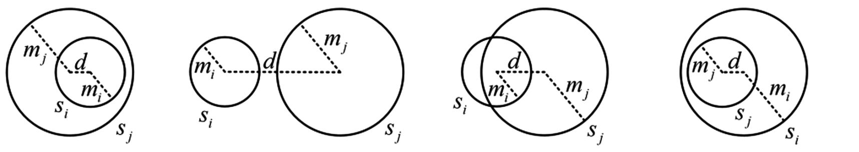
where $\mathbf{E}^{k \times n}$ is an identity matrix used to initialize the mapping matrix. As can be seen from Eq. (1), each mapping matrix is determined by an instance and a concept. Hence, TransFG uses different matrices to project the same instance in different `instanceOf` triples, and the projected vectors are also different. The projected vector $\mathbf{i}_\perp$ of $\mathbf{i}$ is defined as Eq. (2).

$$\mathbf{i}_\perp = \mathbf{M}_{ic}\mathbf{i} \tag{2}$$

In TransFG, the `instanceOf` relations are represented using the relative positions between the projected vectors and the concept spheres. For an `instanceOf` triple (i, r_e, c), if it is a positive triple, then $\mathbf{i}_\perp$ should be inside the concept sphere of c to represent the `instanceOf` relation between i and c. If $\mathbf{i}_\perp$ is outside the concept sphere, the instance embedding and concept embedding need to be optimized. The loss function is defined as Eq. (3).

$$f_e(i, c) = ||\mathbf{i}_\perp - \mathbf{p}||_2 - m \tag{3}$$

SubClassOf Triple Representation. For a `subClassOf` triple (c_i, r_c, c_j), the concepts c_i and c_j are represented by the spheres $s_i(\mathbf{p}_i, m_i)$ and $s_j(\mathbf{p}_j, m_j)$. There are four relative positions between the two spheres, as illustrated in Fig. 2 (this figure is taken from [13]). As shown in Fig. 2(a) and described in [13], if (c_i, r_c, c_j) is a positive triple, the sphere s_i should be inside the sphere s_j to represent the inclusion relation between the two concepts, which is the optimization goal.



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

Fig. 2. Four relative positions between concept spheres s_i and s_j. (Source: Figure 2 in [13])

If the two spheres are separate from each other (as shown in Fig. 2(b)) or intersect (as shown in Fig. 2(c)), the two spheres need to get closer via optimization. The loss function is thus defined as Eq. (4) [13].

$$f_c(c_i, c_j) = ||\mathbf{p}_i - \mathbf{p}_j||_2 + m_i - m_j \tag{4}$$

where $||\mathbf{p}_i - \mathbf{p}_j||_2$ denotes the distance d between $\mathbf{p}_i$ and $\mathbf{p}_j$ of the two spheres.

If s_j is inside s_i as shown in Fig. 2(d), we need to reduce m_i and increase m_j. The loss function is therefore defined as Eq. (5) [13].

$$f_c(c_i,\ c_j) = m_i - m_j \tag{5}$$

Inter-instance Relation Triple Representation. Just like TransD [9], for an inter-instance relation triple $(h,\ r,\ t)$, TransFG learns six vectors: the head instance vector $\mathbf{h}$, relation vector $\mathbf{r}$, tail instance vector $\mathbf{t}$, projection vector $\mathbf{h}_p$ for h, projection vector $\mathbf{r}_p$ for r, and projection vector $\mathbf{t}_p$ for t. The projection vectors $\mathbf{h}_p$, $\mathbf{r}_p$, and $\mathbf{t}_p$ are used to construct the mapping matrices $\mathbf{M}_{rh}$ and $\mathbf{M}_{rt}$, which are defined as Eqs. (6) and (7) [9].

$$\mathbf{M}_{rh} = \mathbf{r}_p\mathbf{h}_p^\top + \mathbf{E}^{z\times n} \tag{6}$$

$$\mathbf{M}_{rt} = \mathbf{r}_p\mathbf{t}_p^\top + \mathbf{E}^{z\times n} \tag{7}$$

where $\mathbf{E}^{z\times n}$ is an identity matrix. TransFG projects $\mathbf{h}$ and $\mathbf{t}$ from the instance space to the corresponding relation space through the mapping matrices, obtaining the projected vectors $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$. The vectors $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ are defined as Eq. (8) [9]:

$$\mathbf{h}_\perp = \mathbf{M}_{rh}\mathbf{h},\quad \mathbf{t}_\perp = \mathbf{M}_{rt}\mathbf{t} \tag{8}$$

The loss function is then defined as Eq. (9) [9].

$$f_r(h,\ t) = \ ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2 \tag{9}$$

Finally, similar to other embedding models [6–9, 13], we enforce constraints as $||\mathbf{h}||_2 \le 1$, $||\mathbf{t}||_2 \le 1$, $||\mathbf{r}||_2 \le 1$, $||\mathbf{p}||_2 \le 1$, $||\mathbf{h}_\perp||_2 \le 1$, $||\mathbf{t}_\perp||_2 \le 1$ in our experiments.

2.3 Model Training

The KG contains only positive triples, but model training requires negative triples, which need to be created with positive triples. For a positive triple $(s,\ p,\ o)$ in the training set, either negative triple $(s',\ p,\ o)$ or negative triple $(s,\ p,\ o')$ is generated by replacing s or o with the same type of KG element (instance or concept) as s or o. Like many existing studies, we use two replacement strategies, "unif" and "bern" [7], to generate negative triples. The replacement strategy "unif" means replacing the subjects or the objects in positive triples with the same probability, while "bern" means replacing the subjects or the objects with the different probabilities for reducing false negative labels. Each positive or negative triple is indicated by a label.

Just like TransC [13], we define the margin-based ranking loss $\mathcal{L}_e$ for the `instanceOf` triples as Eq. (10) [13].

$$\mathcal{L}_e = \sum_{\xi\in\mathcal{S}_e}\sum_{\xi'\in\mathcal{S}'_e} \max(0,\ \gamma_e + f_e(\xi) - f_e(\xi')) \tag{10}$$

Similarly, the margin-based ranking loss $\mathcal{L}_c$ for the `subClassOf` triples and the margin-based ranking loss $\mathcal{L}_l$ for inter-instance relation triples are defined as Eqs. (11) and (12) [13].

$$\mathcal{L}_c = \sum_{\xi \in \mathcal{S}_c} \sum_{\xi' \in \mathcal{S}'_c} \max(0,\ \gamma_c + f_c(\xi) - f(\xi')) \tag{11}$$

$$\mathcal{L}_l = \sum_{\xi \in \mathcal{S}_l} \sum_{\xi' \in \mathcal{S}'_l} \max(0,\ \gamma_l + f_r(\xi) - f_r(\xi')) \tag{12}$$

The overall ranking loss $\mathcal{L}$ is therefore defined as Eq. (13) [13].

$$\mathcal{L} = \mathcal{L}_e + \mathcal{L}_c + \mathcal{L}_l \tag{13}$$

The goal of model training is to minimize the overall ranking loss using stochastic gradient descent (SGD).

3 Experimental Evaluation

3.1 Experimental Design

Evaluation Tasks. We used two typical KG downstream tasks, triple classification and link prediction, to compare and evaluate our TransFG and several KG embedding models including TransC [13], TransE [6], TransH [7], TransR [8], TransD [9], HolE [10], DistMult [11], and ComplEx [12]. We used Accuracy, Precision, Recall and F1-score as the evaluation metrics for the triple classification task, while we used MRR and Hits@N as the evaluation metrics for the link prediction task.

Implementation. The program code of TransC directly uses its **C++** code published in [13] (cf. https://github.com/davidlvxin/TransC). The program code of TransFG was generated by modifying the loss function calculation module and model training module of TransC's code. These codes were used to generate the corresponding experimental results. We copied the results of the other models from [13], as both experiments used the same experimental datasets and parameter settings.

Datasets. We used the same experimental datasets YAGO39K and M-YAGO39K as in [13]. YAGO39K was built in [13] by randomly extracting triples from YAGO, consisting of 39 types of relations including `instanceOf`, `subClassOf`, and inter-instance relations. As stated in [13], M-YAGO39K was formed based on YAGO39K by generating some new triples using the transitivity of the IS-A relations. We trained the TransC and TransFG models using the training set of YAGO39K, and obtained the best parameter configurations on YAGO39K and on M-YAGO39K through the validation sets of the two datasets, respectively. The triple classification task was evaluated on the test sets of the two datasets, respectively, while the link prediction task was evaluated on the test set of YAGO39K.

3.2 Experimental Results of Triple Classification

Triple classification is a binary classification task that determines whether a given triple is a positive triple or not. When the triple classification task is performed by TransFG, we set a threshold δ_r for each relation r. The threshold is achieved by maximizing the classification accuracy on the validation set. For each triple in the test set, TransFG uses the loss function defined for the triple to calculate the score. If the score is less than the threshold, the triple is classified as positive; otherwise it is classified as negative.

In our experiments, we set the parameters of TransC according to the best configurations given in [13]. For TransFG, we select the learning rate for SGD among $\{0.1, 0.01, 0.001\}$, the dimension of concept, instance and relation vectors k, n, z among $\{20, 50, 100\}$, and the margins γ_e, γ_c, γ_l among $\{0.1, 0.3, 0.5, 1\}$. The best configurations of TransFG are then determined by the classification accuracy on the validation set. The best configurations on both YAGO39K and M-YAGO39K datasets are: $\lambda = 0.001$, $k = 100$, $n = 100$, $z = 100$, $\gamma_e = 0.1$, $\gamma_c = 0.1$, $\gamma_l = 1$, and taking L_1 as dissimilarity. We train the TransC and TransFG models for 1,000 rounds.

The experimental results of performing the `instanceOf`, `subClassOf`, and inter-instance relation triple classification tasks on the experimental datasets are shown in Tables 2, 3, and 4, respectively, where "P" stands for Precision, "R" Recall, and "F1" F1-Score, and "unif" and "bern" are the two replacement strategies described earlier. Observing these results, we have the following findings:

1. From Table 2, we can find that TransC and TransFG perform slightly worse than other models on the `instanceOf` triple classification task on YAGO39K. We agree with the viewpoint in [13]: Since the `instanceOf` triples account for the majority (53.5%) in YAGO39K, the `instanceOf` relation is trained too many times in the other models, resulting in almost the best performance on this task, but the performance on other triple classification tasks is not good. It is worth noting that TransC and TransFG achieve relatively balanced performance on all three triple classification tasks. TransFG achieves slightly worse performance than TransC under the "unif" strategy, but it achieves better performance than TransC under the "bern" strategy.

Table 2. Results (%) of the `instanceOf` triple classification on the two datasets.

Model	YAGO39K				M-YAGO39K			
	Accuracy	P	R	F1	Accuracy	P	R	F1
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)	80.3	81.6	80.0	80.8	85.5↑	88.4↑	81.9↑	85.0↑
TransC (bern)	79.8	83.3	74.5	78.6	85.4↑	86.2↑	**84.3**↑	85.2↑
TransFG (unif)	80.2	82.4	78.6	80.4	85.5↑	88.3↑	82.0↑	85.0↑
TransFG (bern)	81.7	83.8	75.5	79.4	**85.9**↑	**88.4**↑	82.2↑	**85.2**↑

Table 3. Results (%) of the `subClassOf` triple classification on the two datasets.

Model	YAGO39K				M-YAGO39K			
	Accuracy	P	R	F1	Accuracy	P	R	F1
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)	83.0	77.2	93.7	84.6	83.1↑	77.6↑	93.2↓	84.7↑
TransC (bern)	83.8	78.1	93.9	85.3	84.5↑	**80.8**↑	90.4↓	85.3↑
TransFG (unif)	82.8	75.7	**96.5**	84.8	83.1↑	76.5↑	**95.5**↓	84.9↑
TransFG (bern)	**84.5**	**78.6**	95.2	**86.1**	**84.7**↑	78.7↑	94.1↓	**85.7**↓

Table 4. Results (%) of inter-instance relation triple classification on YAGO39K.

Model	YAGO39K			
	Accuracy	P	R	F1
TransE	92.1	92.8	91.2	92.0
TransH	90.8	91.2	90.3	90.8
TransR	91.7	91.6	91.9	91.7
TransD	89.3	88.1	91.0	89.5
HolE	92.3	92.6	91.9	92.3
DistMult	93.5	93.9	93.0	93.5
ComplEx	92.8	92.6	93.1	92.9
TransC (unif)	93.5	94.4	92.7	93.6
TransC (bern)	93.8	**94.9**	92.7	93.8
TransFG (unif)	93.1	94.0	92.7	93.4
TransFG (bern)	**94.4**	94.7	**93.3**	**94.0**

2. From Tables 2, 3, and 4, we can find that TransFG performs better than TransC on all three triple classification tasks in terms of almost all evaluation metrics, except for the Recall value of the `instanceOf` triple classification task on M-YAGO39K, the Precision value of the `subclassOf` triple classification task on M-YAGO39K, and the Precision value of the inter-instance triple classification task on YAGO39K. This suggests that TransFG does better than TransC in terms of the representation of multiple different properties possessed by an instance and the modeling of the `instanceOf` relations. TransFG also performs better than all other models on all three triple classification tasks in terms of all evaluation metrics, except for the `instanceOf` triple classification task on YAGO39K.
3. From Tables 2 and 3, we can find that in most cases of performing the `instanceOf` and `subClassOf` triple classification tasks, TransFG performs better

on the M-YAGO39K dataset than on the YAGO39K dataset, which indicates that TransFG can handle the transitivity of the IS-A relations very well.
4. From Tables 2, 3, and 4, we can find that in most cases of performing all three triple classification tasks on the both datasets, TransFG performs better under the "bern" strategy than under the "unif" strategy, which indicates that the "bern" strategy can reduce false negative labels more effectively than the "unif" strategy.

Based on the above findings, we can conclude that the performance of TransFG performing the triple classification task is generally better than that of TransC and other compared models.

3.3 Experimental Results of Link Prediction

Link prediction is aimed at predicting the missing head or tail for an inter-instance triple. When the link prediction task is performed by TransFG, for each triple in the test set we first replace the head instance and the tail instance with all instance in $\mathcal{I}$ one by one, thereby obtaining so-called *corrupted triples* (two corrupted triples per replacement). Then we obtain the scores by calculating the loss function defined for the corrupted triples, and finally rank the instances in $\mathcal{I}$ in ascending order of the scores. Note that a corrupted triple may also exist in the KG, so such a triple that exists in the KG should be regarded as a correct prediction (a positive triple). Like existing works [6–10, 12, 13], our experiments also use two common evaluation settings "Raw" and "Filter". The "Raw" setting means that the corrupted but positive triples are not filtered out, while the "Filter" setting means that these triples are filtered out.

In our experiments, we set the parameters of TransC according to the best configurations given in [13]. For TransFG, the parameters are selected in the same way as on the triple classification task as described in Sect. 3.2. The best configurations of TransFG are then determined according to the Hits@10 on the verification set. The best configurations on the YAGO39K dataset are: $\lambda = 0.001$, $k = 100$, $n = 100$, $z = 100$, $\gamma_e = 0.1$, $\gamma_c = 1$, $\gamma_l = 1$, and taking L_1 as dissimilarity. We train the TransC and TransFG models for 1,000 rounds.

The experimental results of performing the link prediction task on YAGO39K are shown in Table 5. Observing these results, we have the following findings:

1. TransFG performs slightly worse than DisMult, but is the same as TransE and better than all other models, in terms of MRR (in the "Raw" setting).
2. TransFG outperforms TransC and all other models in terms of MRR (in the "Filter" settings) and Hits@N, which indicates that TransFG can represent multiple different properties possessed by an instance very well.
3. TransFG performs better under the "bern" strategy than under the "unif" strategy, which indicates that the "bern" strategy can reduce false negative labels more effectively than the "unif" strategy.

Based on the above findings, we can conclude that the performance of TransFG performing the link prediction task is generally better than that of TransC and other compared models.

Table 5. Results of link prediction for inter-instance relation triples on YAGO39K. Hist@N uses the results in the "Filter" evaluation setting.

Model	YAGO39K				
	MRR		Hits@N (%)		
	Raw	Filter	1	3	10
TransE	0.114	0.248	12.3	28.7	51.1
TransH	0.102	0.215	10.4	24.0	45.1
TransR	0.112	0.289	15.8	33.8	56.7
TransD	0.113	0.176	8.9	19.0	35.4
HolE	0.063	0.198	11.0	23.0	38.4
DisMult	**0.156**	0.362	22.1	43.6	66.0
ComplEx	0.058	0.362	29.2	40.7	48.1
TransC (unif)	0.087	0.421	28.3	50.0	69.2
TransC (bern)	0.112	0.420	29.8	50.2	69.8
TransFG (unif)	0.105	0.404	28.3	50.2	69.4
TransFG (bern)	0.114	**0.475**	**32.5**	**52.1**	**70.1**

4 Conclusions

Based on TransC, this paper proposes a fine-grained KG embedding model TransFG that embeds concepts, instances, and relations into different vector spaces and projects instance vectors from the instance space to the concept space and the relation spaces through dynamic mapping matrices. We conducted experimental evaluation through link prediction and triple classification on datasets YAGO39K and M-YAGO39K. The results show that TransFG outperforms TransC and other typical KG embedding models in most cases, especially in terms of the representation of multiple different properties possessed by an instance and the modeling of the `instanceOf` relations.

References

1. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
2. Rosso, P., Yang, D., Cudré-Mauroux, P.: Knowledge graph embeddings. In: Sakr, S., Zomaya, A.Y. (eds.) Encyclopedia of Big Data Technologies. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-63962-8_284-1
3. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 697–706. ACM (2007). https://doi.org/10.1145/1242572.1242667
4. Lehmann, J., Isele, R., Jakob, M., et al.: DBpedia - a large-scale, multilingual knowledge base extracted from Wikipedia. Semant. Web **6**(2), 167–195 (2015)
5. Li, W., Chai, L., Yang, C., Wang, X.: An evolutionary analysis of DBpedia datasets. In: Meng, X., Li, R., Wang, K., Niu, B., Wang, X., Zhao, G. (eds.) WISA 2018. LNCS, vol. 11242, pp. 317–329. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02934-0_30

6. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, NIPS 2013, pp. 2787–2795 (2013). http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data
7. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2014, pp. 1112–1119. AAAI Press (2014). https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8531
8. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015, pp. 2181–2187. AAAI Press (2015). https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9571
9. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, vol. 1, pp. 687–696. The Association for Computer Linguistics (2015). https://www.aclweb.org/anthology/P15-1067
10. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 1955–1961. AAAI Press (2016). https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12484
11. Yang, B., Yih, W.-T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: 3rd International Conference on Learning Representations, ICLR 2015. Conference Track Proceedings (2015). https://arxiv.org/pdf/1412.6575
12. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, JMLR Workshop and Conference Proceedings, vol. 48, pp. 2071–2080. JMLR.org (2016). http://proceedings.mlr.press/v48/trouillon16.html
13. Lv, X., Hou, L., Li, J., Liu, Z.: Differentiating concepts and instances for knowledge graph embedding. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, pp. 1971–1979. Association for Computational Linguistics (2018). https://aclweb.org/anthology/papers/D/D18/D18-1222/