

# KGE-CL: Contrastive Learning of Tensor Decomposition Based Knowledge Graph Embeddings

Zhiping Luo<sup>1,4\*</sup>, Wentao Xu<sup>1,4\*</sup>, Weiqing Liu<sup>2</sup>, Jiang Bian<sup>2</sup>, Jian Yin<sup>3,4†</sup>, and Tie-Yan Liu<sup>2</sup>

<sup>1</sup> School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China

<sup>2</sup> Microsoft Research Asia, Beijing, China

<sup>3</sup> School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China

<sup>4</sup> Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China

{luozhp7@mail2, xuwt6@mail2, issjyin@mail}.sysu.edu.cn

{weiqing.liu, jiang.bian, tyliu}@microsoft.com

## Abstract

Learning the embeddings of knowledge graphs (KG) is vital in artificial intelligence, and can benefit various downstream applications, such as recommendation and question answering. In recent years, many research efforts have been proposed for knowledge graph embedding (KGE). However, most previous KGE methods ignore the semantic similarity between the related entities and entity-relation couples in different triples since they separately optimize each triple with the scoring function. To address this problem, we propose a simple yet efficient contrastive learning framework for tensor decomposition based (TDB) KGE, which can shorten the semantic distance of the related entities and entity-relation couples in different triples and thus improve the performance of KGE. We evaluate our proposed method on three standard KGE datasets: WN18RR, FB15k-237 and YAGO3-10. Our method can yield some new state-of-the-art results, achieving 51.2% MRR, 46.8% Hits@1 on the WN18RR dataset, 37.8% MRR, 28.6% Hits@1 on FB15k-237 dataset, and 59.1% MRR, 51.8% Hits@1 on the YAGO3-10 dataset. Source codes and data of this paper can be found at <https://github.com/Wentao-Xu/KGE-CL>.

## 1 Introduction

The knowledge graph (KG) stores a vast number of human knowledge in the format of triples. A triple  $(h, r, t)$  in a KG contains a head entity  $h$ , a tail entity  $t$ , and a relation  $r$  between  $h$  and  $t$ . The knowledge graph embedding (KGE) aims to project the massive interconnected entities and relations in a KG into vectors or matrices, which can preserve the semantic information of the triples. Learning the embeddings of KG can benefit various downstream artificial intelligence applications, such as

question answering (Huang et al., 2019), machine reading comprehension (Yang and Mitchell, 2017), image classification (Marino et al., 2016), and personalized recommendation (Wang et al., 2018).

In general, most of the KGE methods would define a scoring function  $f(h_i, r_j, t_k)$ , and the training target of KGE is maximizing the score of a true triple  $(h_i, r_j, t_k)$  and minimizing the score of a false triple  $(h_i, r_j, t_x)$ . In this way, the trained embeddings of entities and relations in KG can preserve the intrinsic semantics of a true triple. We can mainly divide the existing KGE methods into two categories. The first category is the distance based (DB) methods, which use the Minkowski distance as scoring function to measure a triple’s plausibility, include the TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015), TransD (Ji et al., 2015) and TransG (Xiao et al., 2016). The other category is the tensor decomposition based (TDB) methods, which treat a KG as a third-order binary tensor and use the results of tensor decomposition as the representations of entities and relations. The TDB methods include the CP (Hitchcock, 1927), DistMult (Yang et al., 2015), RESCAL (Nickel et al., 2011) and ComplEx (Trouillon et al., 2016).

However, existing methods only capture the semantic connections among  $h$ ,  $r$  and  $t$  in a same triple. For example, the TransE optimize the distance between  $h + r$  and  $t$ , and the DistMult optimizie the dot product similarity among  $h$ ,  $r$  and  $t$ . Therefore, they overlook the connections between the related entities and entity-relation couples in different triples. In a KG, some entities (entity-relation couples) that share the same entity-relation couple (entity) are in the same type and have semantic similarity. Capturing the semantic similarity of these entities or couples can improve the expressiveness of embeddings, which is the performance in capturing semantic information in KG (Dettmers et al., 2018; Xu et al.,

\*The first two authors contributed equally.

†Corresponding author.

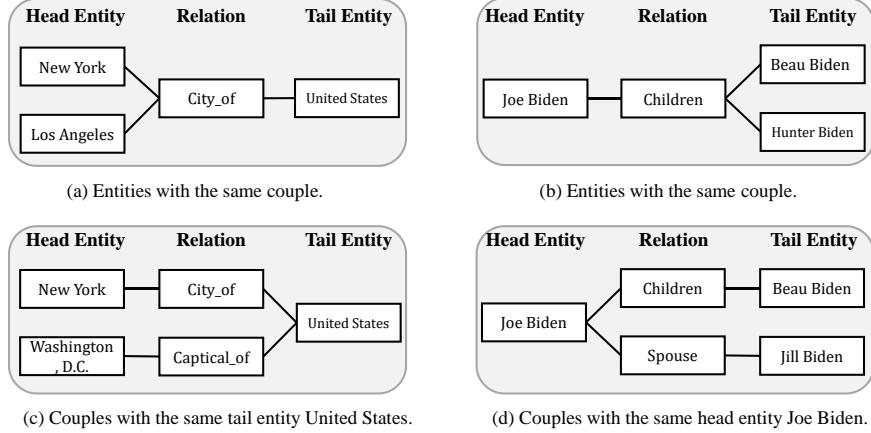


Figure 1: The examples of triples that share the same entities or entity-relation couples.

2020). For instance, in Figure 1 (a), the entities *New York* and *Los Angeles* share the same entity-relation couple (*City\_of*, *United States*). Therefore, the entities *New York* and *Los Angeles* should have a similar embedding. Besides, in Figure 1 (c) the couples (*New York*, *City\_of*) and (*Washington, D.C.*, *Captical\_of*) share the same tail entity *United States*. Thus the representations of these two couples should also be similar.

To correlate the related entities and entity-relation couples in different triples, we propose a simple yet efficient contrastive learning framework called KGE-CL for KGE, which is quite general for existing TDB methods. We first construct the positive instances for those entities that share the same entity-relation couple and those entity-relation couples that share the same entity. For example, the positive instance of *Beau Biden* in Figure 1 (b) is the *Hunter Biden*, and the positive instance of (*Children*, *Beau Biden*) in Figure 1 (d) is (*Spouse*, *Jill Biden*). Then we calculate the contrastive loss on the original instance and the positive instances. Due to the design of the contrastive learning framework, we can also increase the distance between unrelated entities and couples. Finally, since each triple has four positive instances (corresponding to the four examples in Figure 1), we design a weighted contrastive loss to control the weights on different positive instances' loss flexibly.

We evaluate our KGE-CL method on the KG link prediction task using the standard WN18RR (Toutanova and Chen, 2015), FB15k-237 (Dettmers et al., 2018) and YAGO3-10 (Mahdisoltani et al., 2015) datasets. Our proposed method achieves new state-of-the-art results (SotA), obtaining 51.2% MRR, 46.8% Hits@1 on

the WN18RR dataset, 37.8% MRR, 28.6% Hits@1 on the FB15k-237 dataset, and 59.1% MRR, 51.8% Hits@1 on the YAGO3-10 dataset. Moreover, We apply several experiments to further analyze the inner mechanism of our method. At last, to clearly explain why our method outperforms existing methods, we conduct the visualization of the KGE of our method and some compared methods using T-SNE (van der Maaten and Hinton, 2008).

In summary, this paper's contributions include:

- We propose KGE-CL, a simple yet efficient contrastive learning framework for TDB KGE. It can capture the semantic similarity of the related entities and entity-relation couples in different triples, thus improving the performance of KGE.
- Our proposed KGE-CL framework can also push the embeddings of unrelated entities and couples apart in the semantic space.
- The experiment results and analyses confirm the effectiveness of our KGE-CL method.

## 2 Related Work

In recent years, knowledge graph embedding (KGE) becomes a pretty hot research topic since its vital role in various downstream applications. We can categorize the existing KGE techniques into two categories: the distance based KGE and tensor decomposition based KGE.

Distance based (DB) methods describe relations as relational maps between head and tail entities. The TransE is a representative distance based method, which uses the relations as translations and its scoring function is:  $f(h_i, r_j, t_k) = -\|\mathbf{h}_i +$

$\mathbf{r}_j - \mathbf{t}_k\|_2^2$ . To improve the performance of TransE, many its variants that follow the same direction were proposed, such as the TransH (Wang et al., 2014), TransR (Lin et al., 2015), TransD (Ji et al., 2015), TranSparse (Ji et al., 2016), TransG (Xiao et al., 2016) and RotatE (Sun et al., 2019). However, the TransE and its extensions can not capture the semantic similarity between the related entities and entity-relation couples in different triples. For example, given two triples  $(h_1, r_1, t_1)$  and  $(h_1, r_1, t_2)$ , TransE can only close the distance between  $\mathbf{h}_1 + \mathbf{r}_1$  and  $\mathbf{t}_1$  (or  $\mathbf{t}_2$ ) in the same triple, it does not close the distance between  $\mathbf{t}_1$  and  $\mathbf{t}_2$ , so the representations  $\mathbf{t}_1$  and  $\mathbf{t}_2$  can in different directions.

Tensor decomposition based (TDB) methods formulate the KGE task as a third-order binary tensor decomposition problem. RESCAL (Nickel et al., 2011) factorizes the  $j$ -th frontal slice of  $\mathcal{X}$  as  $\mathcal{X}_j \approx \mathbf{A}\mathbf{R}_j\mathbf{A}^\top$ , in which embeddings of head and tail entities are from the same space. As the relation embeddings in RESCAL are matrices containing lots of parameters, RESCAL is easier to be overfitting and more difficult to train. DistMult (Yang et al., 2015) simplifies the matrix  $\mathbf{R}_j$  in RESCAL to a diagonal matrix, while the RESCAL can only preserve the symmetry of relations, limiting its expressiveness. To model asymmetric relations, ComplEx (Trouillon et al., 2016) extends DistMult to complex embeddings and preserving the relations' symmetry in the real part and the asymmetry in the imaginary part. Moreover, the QuatE (Zhang et al., 2019) further extends the ComplEx to hypercomplex space to model more complicated relation properties, such as the inversion. All of the DistMult, ComplEx, and QuatE are the variants of CP decomposition (Hitchcock, 1927), which are in real, complex, and hypercomplex vector spaces, respectively. On the other hand, the TDB methods usually suffer from an overfitting problem; thus, some work is trying to improve the TDB methods from the aspect of regularizer, such as the N3 (Lacroix et al., 2018) and DURA (Zhang et al., 2020a) regularizers. These regularizers bring more significant improvements than the original squared Frobenius norm ( $L_2$  norm) regularizer (Nickel et al., 2011; Yang et al., 2015; Trouillon et al., 2016). Nevertheless, the TDB methods can only capture the similarity among  $h$ ,  $r$  and  $t$  in a same triple. For example, the ComplEx-DURA (Zhang et al., 2020a) mainly capture the semantics similarity between the couple  $(h_i, r_j)$ 's embedding  $\mathbf{h}_i\mathbf{R}_j$  and tail entity  $t_k$ 's

embedding  $\mathbf{t}_k$  in a sample triple.

Since both of the DB and TDB methods can not correctly capture the semantic similarity between related entities and couples in different triples, we propose our KGE-CL method to address the limitations of existing work.

### 3 Preliminaries

#### 3.1 Knowledge Graph Embedding

**Knowledge Graph Embedding (KGE)** The knowledge graph embedding (KGE) is to learn the representations (may be real or complex vectors, matrices, and tensors) of the entities and relations. Its target is that the learned entities' and relations' embeddings can preserve the semantic information of the triples in knowledge graphs. Generally, the KGE methods define a scoring function  $f(h_i, r_j, t_k)$  to score the corresponding triple  $(h_i, r_j, t_k)$ , and the score measure the plausibility of triples.

**Tensor Decomposition Based (TDB) KGE** TDB methods like RESCAL (Nickel et al., 2011) and ComplEx (Trouillon et al., 2016), regard a KG as a third-order binary tensor  $\mathcal{X} \in \{0, 1\}^{|\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}|}$ . The  $(i, j, k)$  entry  $\mathcal{X}_{ijk} = 1$  if  $(h_i, r_j, t_k)$  is a true triple otherwise  $\mathcal{X}_{ijk} = 0$ . The  $\mathcal{X}_j$  denotes the  $j$ -th frontal slice of  $\mathcal{X}$ , that is, the corresponding matrix of the  $j$ -th relation. Generally, a TDB KGE model factorizes  $\mathcal{X}_j$  as  $\mathcal{X}_j \approx \mathbf{Re}(\mathbf{H}\mathbf{R}_j\bar{\mathbf{T}}^\top)$ , where the  $i$ -th ( $k$ -th) row of  $\mathbf{H}$  ( $\mathbf{T}$ ) is  $\mathbf{h}_i$  ( $\mathbf{t}_k$ ),  $\mathbf{R}_j$  is a matrix that represents relation  $r_j$ ,  $\mathbf{Re}(\cdot)$  and  $\bar{\cdot}$  are the real part and the conjugate of a complex matrix, respectively. Then the scoring functions of TDB KGE methods is:  $f(h_i, r_j, t_k) = \mathbf{Re}(\mathbf{h}_i\mathbf{R}_j\bar{\mathbf{t}}_k^\top)$ . Note that the real part and the conjugate of a real matrix are itself. The goal of TDB models is to seek matrices  $\mathbf{H}, \mathbf{R}_1, \dots, \mathbf{R}_{|\mathcal{R}|}, \mathbf{T}$ , such that  $\mathbf{Re}(\mathbf{H}\mathbf{R}_j\bar{\mathbf{T}}^\top)$  can approximate  $\mathcal{X}_j$ . In this paper, we aim to improve the performance of existing TDB models, such as the RESCAL and ComplEx models.

#### 3.2 Contrastive Learning

Contrastive learning is an efficient representation learning method that contrasts positive pairs against negative pairs (Hadsell et al., 2006; He et al., 2020; Chen et al., 2020; Khosla et al., 2020). The key idea of contrastive learning is pulling the semantically close pairs together and push apart the negative pairs. The unsupervised contrastive learning framework (Chen et al., 2020) would utilize the data augmentation to construct positive pairs to calculate the contrastive loss. The supervised contrastive

learning framework (Khosla et al., 2020) calculates the contrastive loss of all positive instances within the same mini-batch. Motivated by these exiting frameworks, we adopt the following function to calculate the contrastive loss between an instance  $z_i$  and its all positive instances  $z_i^+$ :

$$\text{CL}(\mathbf{z}_i, \mathbf{z}_i^+) = \frac{-\sum_{z_i^+ \in P(i)} \log \frac{e^{\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau}}{\sum_{z_j \in N(i)} e^{\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau}}}{|P(i)|} \quad (1)$$

where  $\mathbf{z}_i$  and  $\mathbf{z}_i^+$  are the representations of  $z_i$  and  $z_i^+$ , respectively.  $P(i)$  is the set of all positive instances in the mini-batch, and  $N(i)$  is the set of all negative instances in the batch. In our work, we define the negative instances as the instances that do not belong to the positive instances.  $\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+) = \mathbf{z}_i \cdot \mathbf{z}_i^+$  is the dot product similarity.

## 4 Our Method: KGE-CL

In this section, we describe our KGE-CL method that utilize the contrastive learning to capture the semantic similarity of related entities and couples in different triples. Our method is very general and can be easily apply to arbitrary TDB methods. We can further name our KGE-CL method as RESCAL-CL or ComplEx-CL when we use the scoring function of RESCAL or ComplEx models. In this section, we firstly present the contrastive loss we designed for the KGE, then we introduce the training objective of our method.

### 4.1 Contrastive Loss of KGE

In this subsection, we elaborate on the contrastive loss that we designed for KGE.

**Positive Instances** The generation of positive instances  $z_i^+$  for the instance  $z_i$  is vital in contrastive learning. Existing work in visual representation learning (Wu et al., 2018; Chen et al., 2020; Chen and He, 2020) used some data augmentation methods, such as cropping, color distortion, and rotation, to take two random transformations of the same images as  $z_i$  and  $z_i^+$ . Meanwhile, in NLP, some work (Wu et al., 2020; Meng et al., 2021) utilized other augmentation techniques like word deletion, reordering, and substitution. However, these data augmentation methods are not proper to the KGE. To capture the interactions between triples in a KG, we design a new approach to construct the positive instances for KGE. For a triple  $(h_i, r_j, t_k)$ , the corresponding scoring function of TDB methods is:

$$\begin{aligned} f(h_i, r_j, t_k) &= \mathbf{Re}(\mathbf{h}_i \mathbf{R}_j \bar{\mathbf{t}}_k^\top) = \mathbf{Re}(\langle \mathbf{h}_i \mathbf{R}_j, \bar{\mathbf{t}}_k \rangle) \\ &= \mathbf{Re}(\langle \mathbf{h}_i, \bar{\mathbf{t}}_k \mathbf{R}_j^\top \rangle). \end{aligned} \quad (2)$$

We define  $\langle \cdot, \cdot \rangle$  as the inner product of two real or complex vectors:  $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \mathbf{v}^\top$ . The  $\mathbf{h}_i \mathbf{R}_j$  and  $\bar{\mathbf{t}}_k \mathbf{R}_j^\top$  are the representations of the entity-relation couples  $(h_i, r_j)$  and  $(r_j, t_k)$ , respectively. The Equation 2 means that we can firstly compute either the  $\mathbf{h}_i \mathbf{R}_j$  or  $\bar{\mathbf{t}}_k \mathbf{R}_j^\top$  in the scoring function. For a head entity  $h_i$ , we define its positive instances  $h_i^+$  as those head entities that share the same relation and tail entity with  $h_i$ . Similarly, we define the tail entity  $t_k$ 's positive instances  $t_k^+$  with those tail entities that share the same head entity and relation with  $t_k$ . For the entity-relation couples  $(h_i, r_j)$  or  $(r_j, t_k)$ , the corresponding positive instances  $(h_i, r_j)^+$  or  $(r_j, t_k)^+$  is those entity-relation couples that share the same tail entity with  $(h_i, r_j)$  or head entity with  $(r_j, t_k)$ . The  $(\mathbf{h}_i \mathbf{R}_j)^+$  and  $(\bar{\mathbf{t}}_k \mathbf{R}_j^\top)^+$  are the representations of positive instances  $(h_i, r_j)^+$  and  $(r_j, t_k)^+$ , respectively. Therefore, given a true triple  $(h_i, r_j, t_k)$ , our method would construct four kinds of positive instances:  $h_i^+, t_k^+, (h_i, r_j)^+$  and  $(r_j, t_k)^+$ , which are corresponding to the four examples in Figure 1. As mentioned in Section 3.2, for an instance  $h_i$ , we will use all of its positive instances  $h_i^+$  in the same mini-batch. Besides, there may be no positive instances in a mini-batch for some tripels. Therefore, we will add a postive instance for each triple in the mini-batch, where the added instance is randomly sampled from training set. We will study the effect of positive instances by experiment in Section 5.3.

**Contrastive Loss** Given a true triple  $(h_i, r_j, t_k)$ , we use the Equation 1 to compute the contrastive loss of four types of positive instances, and the overall contrastive loss for a triple  $(h_i, r_j, t_k)$  is:

$$\begin{aligned} \mathcal{L}_c(h_i, r_j, t_k) &= \text{CL}(\mathbf{h}_i, \mathbf{h}_i^+) + \text{CL}(\mathbf{t}_k, \mathbf{t}_k^+) \\ &\quad + \text{CL}(\mathbf{h}_i \mathbf{R}_j, (\mathbf{h}_i \mathbf{R}_j)^+) + \text{CL}(\bar{\mathbf{t}}_k \mathbf{R}_j^\top, (\bar{\mathbf{t}}_k \mathbf{R}_j^\top)^+). \end{aligned} \quad (3)$$

**Theoretical Analysis** To study how the Equation 3 takes effect, we apply a theoretical analysis from the aspect of gradient. Taking the contrastive loss term  $\text{CL}(\mathbf{h}_i, \mathbf{h}_i^+)$  as an example, the gradient

| Contrastive Loss   | WN18RR |      |      |      | FB15k-237 |      |      |      |
|--|--------|------|------|------|-----------|------|------|------|
|  | MRR    | H@1  | H@3  | H@10 | MRR       | H@1  | H@3  | H@10 |
| CL( $\mathbf{h}_i, \mathbf{h}_i^+$ )   | .509   | .465 | .527 | .588 | .376      | .284 | .412 | .557 |
| CL( $\mathbf{t}_k, \mathbf{t}_k^+$ )   | .509   | .468 | .425 | .587 | .376      | .285 | .411 | .558 |
| CL( $\mathbf{h}_i \mathbf{R}_j, (\mathbf{h}_i \mathbf{R}_j)^+$ )                       | .512   | .469 | .527 | .595 | .374      | .282 | .410 | .557 |
| CL( $\bar{\mathbf{t}}_k \mathbf{R}_j^\top, (\bar{\mathbf{t}}_k \mathbf{R}_j^\top)^+$ ) | .504   | .462 | .516 | .580 | .378      | .286 | .414 | .559 |

Table 1: Link prediction results of RESCAL-CL that merely uses one specific contrastive loss term.

of  $\text{CL}(\mathbf{h}_i, \mathbf{h}_i^+)$  to embedding  $\mathbf{h}_i$  is:

$$\begin{aligned} \frac{\partial \text{CL}(\mathbf{h}_i, \mathbf{h}_i^+)}{\partial \mathbf{h}_i} &= \\ \sum_{h_i^+ \in P(i)} \partial \left( \mathbf{h}_i \cdot \mathbf{h}_i^+ / \tau - \log \sum_{h_j \in N(i)} e^{(\mathbf{h}_i \cdot \mathbf{h}_j / \tau)} \right) \\ &\quad - |P(i)| \partial \mathbf{h}_i \\ &= -\frac{\sum_{h_i^+ \in P(i)} \mathbf{h}_i^+}{\tau |P(i)|} + \frac{\sum_{h_j \in N(i)} (e^{(\mathbf{h}_i \cdot \mathbf{h}_j / \tau)} \mathbf{h}_j)}{\tau \sum_{h_j \in N(i)} e^{(\mathbf{h}_i \cdot \mathbf{h}_j / \tau)}}. \end{aligned} \quad (4)$$

Then when we update the embedding  $\mathbf{h}_i$  with the gradient  $\frac{\partial \text{CL}(\mathbf{h}_i, \mathbf{h}_i^+)}{\partial \mathbf{h}_i}$ :

$$\mathbf{h}_i^{t+1} = \mathbf{h}_i^t - \eta \frac{\partial \text{CL}(\mathbf{h}_i, \mathbf{h}_i^+)}{\partial \mathbf{h}_i}, \quad (5)$$

where  $\eta$  of learning rate for updating gradient. The Equation 4 and Equation 5 show that the embedding  $\mathbf{h}_i^t$  would update to the direction of  $\frac{\eta \sum_{h_i^+ \in P(i)} \mathbf{h}_i^+}{\tau |P(i)|}$ , which is the mean value of positive instances' embeddings  $\mathbf{h}_i^+$ . Meanwhile, the  $\mathbf{h}_i^t$  would also update away from the weighted value  $\frac{\eta \sum_{h_j \in N(i)} (e^{(\mathbf{h}_i \cdot \mathbf{h}_j / \tau)} \mathbf{h}_j)}{\tau \sum_{h_j \in N(i)} e^{(\mathbf{h}_i \cdot \mathbf{h}_j / \tau)}}$  of negative instances  $\mathbf{h}_j$ . So our proposed contrastive loss  $\mathcal{L}_c$  can not only pull the related entities and entity-relation couples together in the semantic space but also push the unrelated entities and couples apart.

**Weighted Contrastive Loss** There are four contrastive loss terms in Equation 3. We found that different contrastive loss terms have different effects on different knowledge graphs during our research process. This phenomenon happens may because different knowledge graphs have diverse graph properties, such as the ratio of the number of entities to the number of relations, the number of triples compared with the number of entities. Table 1 shows the results of RESCAL-CL that merely uses one specific contrastive loss term. In WN18RR dataset, using the term  $\text{CL}(\mathbf{h}_i \mathbf{R}_j, (\mathbf{h}_i \mathbf{R}_j)^+)$  achieves the highest results,

while in FB15k-237 dataset,  $\text{CL}(\bar{\mathbf{t}}_k \mathbf{R}_j^\top, (\bar{\mathbf{t}}_k \mathbf{R}_j^\top)^+)$  is the best. Hence, we introduce a weighted contrastive loss, assigning a weight  $\alpha_*$  for each contrastive loss term, and  $\alpha_*$  is a hyper-parameter that can be flexibly tuned for a specific KG. The contrastive loss  $\mathcal{L}_c^w(h_i, r_j, t_k)$  after adding weights  $\alpha_*$  is:

$$\begin{aligned} \mathcal{L}_c^w(h_i, r_j, t_k) &= \alpha_h \text{CL}(\mathbf{h}_i, \mathbf{h}_i^+) + \alpha_t \text{CL}(\mathbf{t}_k, \mathbf{t}_k^+) \\ &\quad + \alpha_{hr} \text{CL}(\mathbf{h}_i \mathbf{R}_j, (\mathbf{h}_i \mathbf{R}_j)^+) \\ &\quad + \alpha_{tr} \text{CL}(\bar{\mathbf{t}}_k \mathbf{R}_j^\top, (\bar{\mathbf{t}}_k \mathbf{R}_j^\top)^+). \end{aligned} \quad (6)$$

## 4.2 Training Objective

Given a training triple  $(h_i, r_j, t_k)$  in a KG, the instantaneous loss of our framework on this triple is:

$$\mathcal{L}(h_i, r_j, t_k) = \mathcal{L}_s + \mathcal{L}_r + \mathcal{L}_c^w, \quad (7)$$

where  $\mathcal{L}_s$  is the loss that measures the discrepancy between scoring function's output  $f(h_i, r_j, t_k)$  and the label  $\mathcal{X}_{ijk}$ .  $\mathcal{L}_r$  is the regularizer, and  $\mathcal{L}_c^w$  is the weighted contrastive loss we introduced in Section 4.1. It should be noted that the additionally added positive instances in the mini-batch is only used to calculate the contrastive loss  $\mathcal{L}_c^w$  and would not be used to calculate the  $\mathcal{L}_s$  and  $\mathcal{L}_r$  losses.

**$\mathcal{L}_s$  Loss** Many previous efforts used the ranking losses (Bordes et al., 2013), binary logistic regression (Trouillon et al., 2016) or sampled multiclass log-loss (Kadlec et al., 2017) to calculate the distance between the scoring function's output and the triple's label. Since (Lacroix et al., 2018) had verified the competitiveness of the full multiclass log-loss, we utilize it as the  $\mathcal{L}_s$  loss in Equation 7.

**Regularizer** Most of the previous work use the squared Frobenius norm ( $L_2$  norm) regularizer (Nickel et al., 2011; Yang et al., 2015; Trouillon et al., 2016) in their object functions. More recently, some work proposed more efficient regularizers to prevent the overfitting of KGE, such as the N3 (Lacroix et al., 2018) and DURA (Zhang et al., 2020a) regularizers. Since the (Zhang et al., 2020a) had shown that DURA regularizer outperforms L2 and N3 regularizers, we use the DURA as the regularizer  $\mathcal{L}_r$  in our work.

## 5 Experiment

In this section, we present thorough empirical studies to evaluate and analyze our proposed framework. We first introduce the experimental setting.

|           | WN18RR | FB15k-237 | YAGO3-10  |
|-----------|--------|-----------|-----------|
| #Entity   | 40,943 | 14,541    | 123,182   |
| #Relation | 11     | 237       | 37        |
| #Train    | 86,835 | 272,115   | 1,079,040 |
| #Valid    | 3,034  | 17,535    | 5,000     |
| #Test     | 3,134  | 20,466    | 5,000     |

Table 2: Statistics of the datasets.

Then we evaluate our framework’s performance on the task of link prediction. Besides, we further analyze the details of our promotion by comparing our method with a baseline on the triples with different relations, and we also study the effect of positive instances to our framework. Finally, we visualize the embeddings of our method and some baselines to explain why our method outperforms baselines.

### 5.1 Exeprimental Setting

**Dataset** We use three standard KG datasets—WN18RR (Toutanova and Chen, 2015), FB15k-237 (Dettmers et al., 2018), and YAGO3-10 (Mahdissoltani et al., 2015) to evaluate the performance of KGE. We divide the datasets into training, validating, and testing sets using the same way of previous work. Table 2 shows the statistics of these datasets. WN18RR, FB15k-237, and YAGO3-10 are extracted from WN18 (Bordes et al., 2013), FB15k (Bordes et al., 2013), and YAGO3 (Mahdissoltani et al., 2015), respectively. Some previous work (Toutanova and Chen, 2015; Dettmers et al., 2018) indicated the test set leakage problem in WN18 and FB15k, where some test triplets may appear in the training dataset in the form of reciprocal relations. Therefore, they suggested using the WN18RR and FB15k-237 datasets to avoid the test set leakage problem.

**Compared Methods** We compare our KGE-CL method with existing state-of-the-art KGE methods, including CP (Hitchcock, 1927), RESCAL (Nickel et al., 2011), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), RotatE (Sun et al., 2019), MuRP (Balazevic et al., 2019), HAKE (Zhang et al., 2020b), ComplEx-N3 (Lacroix et al., 2018), ROTH (Chami et al., 2020), REFE (Chami et al., 2020), CP-DURA (Zhang et al., 2020a), RESCAL-DURA (Zhang et al., 2020a) and ComplEx-DURA (Zhang et al., 2020a).

| Datasets         | Methods    | $d$  | $m$  | $\tau$ | $\alpha_h$ | $\alpha_t$ | $\alpha_{hr}$ | $\alpha_{tr}$ |
|------------------|------------|------|------|--------|------------|------------|---------------|---------------|
| <b>WN18RR</b>    | RESCAL-CL  | 512  | 512  | 0.9    | 0          | 0          | 2.0           | 0             |
|                  | ComplEx-CL | 2000 | 2048 | 0.5    | 0          | 0          | 0             | 2.0           |
| <b>FB15k-237</b> | RESCAL-CL  | 512  | 512  | 0.9    | 0          | 0          | 0             | 2.0           |
|                  | ComplEx-CL | 2000 | 2048 | 0.5    | 2.0        | 0          | 0             | 0             |
| <b>YAGO3-10</b>  | RESCAL-CL  | 512  | 512  | 0.9    | 0          | 0          | 0             | 1.0           |
|                  | ComplEx-CL | 2000 | 2048 | 0.5    | 0          | 1.0        | 0             | 0             |

Table 3: The selection of the hyper-parameters of RESCAL-CL and ComplEx-CL on different datasets.

**Implementation Details** We implement our method base on the PyTorch library (Paszke et al., 2019), and run on all experiments with a single NVIDIA Tesla V100 GPU. We leverage Adagrad algorithm (Duchi et al., 2011) to optimize the objective function in Equation 7. We tune our model using the grid search to select the optimal hyper-parameters based on the performance on the validation dataset. We search the embedding size  $d$  in {256, 512, 1024} for RESCAL-CL and {200, 500, 1000, 2000} for ComplEx-CL. We search the temperature  $\tau$  in Equation 1 in {0.3, 0.5, 0.7, 0.9, 1.0}. We search the weights  $\alpha_h$ ,  $\alpha_t$ ,  $\alpha_{hr}$  and  $\alpha_{tr}$  in Equation 6 in {0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, 2.5}. The best choices of hyper-parameters, the number of parameters, and the training time of RESCAL-CL and ComplEx-CL on each dataset are listed in Table 3. For a fair comparison, the RESCAL-CL and ComplEx-CL have the same embedding size as RESCAL-DURA and ComplEx-DURA, respectively. Besides, the batch size is 512 for RESCAL-CL and 200 for ComplEx-CL, and the learning rate  $\eta$  as 0.1 for all methods. On WN18RR, we set the number of training epochs as 50 for the ComplEx-CL and 200 for the RESCAL-CL. On FB15k-237 and YAGO3-10, the number of training epochs is 200 for all methods.

### 5.2 Main Results

We evaluate the performance of our framework on the link prediction task, which is a frequently-used task to evaluate the KGE. Specifically, we replace the head or tail entity of a true triple in the test set with other entities in the dataset and name these derived triples as *corrupted triples*. The link prediction task aims to score the original true triples higher than the corrupted ones. We rank the triples by the results of the scoring function.

The evaluation metrics we used in the link prediction are the MRR and Hits@N: 1) MRR: the mean reciprocal rank of original triples; 2) Hits@N: the percentage rate of original triples ranked at the top

| Methods           | WN18RR      |             |             |             | FB15k-237   |             |             |             | YAGO3-10    |             |             |             |
|-------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
|                   | MRR         | H@1         | H@3         | H@10        | MRR         | H@1         | H@3         | H@10        | MRR         | H@1         | H@3         | H@10        |
| CP                | .438        | .414        | .445        | .485        | .333        | .247        | .363        | .508        | .567        | .494        | .611        | .698        |
| RESCAL            | .455        | .419        | .461        | .493        | .353        | .264        | .385        | .528        | .566        | .490        | .612        | .701        |
| ComplEx           | .460        | .428        | .473        | .522        | .346        | .256        | .386        | .525        | .573        | .500        | .617        | .703        |
| ConvE             | .43         | .40         | .44         | .52         | .325        | .237        | .356        | .501        | .44         | .35         | .49         | .62         |
| RotatE            | .476        | .428        | .492        | .571        | .338        | .241        | .375        | .533        | .495        | .402        | .550        | .670        |
| MuRP              | .481        | .440        | .495        | .566        | .335        | .243        | .367        | .518        | -           | -           | -           | -           |
| HAKE              | .497        | .452        | .516        | .582        | .346        | .250        | .381        | .542        | .546        | .462        | .596        | .694        |
| ComplEx-N3        | .491        | .448        | .505        | .580        | .366        | .271        | .403        | .558        | .577        | .502        | .619        | .711        |
| ROTH              | .496        | .449        | .514        | .586        | .344        | .246        | .380        | .535        | .570        | .495        | .612        | .706        |
| REFE              | .473        | .430        | .485        | .561        | .351        | .256        | .390        | .541        | .577        | .503        | .621        | .712        |
| CP-DURA           | .478        | .441        | .497        | .552        | .367        | .272        | .402        | .555        | .582        | .511        | .623        | .708        |
| RESCAL-DURA       | .498        | .455        | .514        | .577        | .368        | .276        | .402        | .550        | .579        | .505        | .619        | .712        |
| ComplEx-DURA      | .491        | .449        | .504        | .571        | .371        | .276        | .408        | .560        | .584        | .511        | .628        | .713        |
| <b>RESCAL-CL</b>  | <b>.512</b> | <b>.468</b> | <b>.531</b> | <b>.597</b> | <b>.378</b> | <b>.286</b> | <b>.414</b> | <b>.559</b> | <b>.581</b> | <b>.507</b> | <b>.625</b> | <b>.713</b> |
| <b>ComplEx-CL</b> | .505        | .458        | .522        | .595        | .377        | .285        | <b>.414</b> | <b>.564</b> | <b>.591</b> | <b>.518</b> | <b>.634</b> | <b>.722</b> |

Table 4: Link prediction results on WN18RR, FB15k-237 and YAGO3-10 datasets. We take the results of CP, RESCAL, ComplEx, CP-DURA, RESCAL-DURA and ComplEx-DURA from the paper (Zhang et al., 2020a), and the results of other baselines are from their original papers.

$N$  in prediction. For both metrics, we remove some of the corrupted triples that exist in datasets from the ranking results, which is also called *filtered* setting in (Bordes et al., 2013). For the metrics of Hits@N, we use Hits@1, Hits@3, and Hits@10.

Table 4 shows the results of link prediction on WN18RR, FB15K-237, and YAGO3-10 datasets. Our proposed method achieves the highest results on all datasets compared with the baselines. Specifically, the RESCAL-CL achieves evidently better results on the WN18RR dataset. The ComplEx-CL outperforms the compared methods in the YAGO3-10 dataset. On FB15k-237, the RESCAL-CL and ComplEx-CL are both better than the RESCAL-DURA and ComplEx-DURA, respectively. The results of RESCAL-CL and ComplEx-CL verify that correlating the entities and entity-relation couples in different triples can boost the performance of KGE.

### 5.3 Model Analysis

**Analyzing the Improvements** To further explore why our method outperforms existing state-of-the-art techniques, we compare our RESCAL-CL method with the RESCAL-DURA on the triples with different relations in WN18RR. Table 5 shows the results of the comparison, and we found that our RESCAL-CL is significantly better than the RESCAL-DURA in 9 out of the 11 relations, verifying that the promotion of our framework is extensive and not just on some specific relations.

| Relations          | #Train | #Test | RESCAL-DURA |       |       | RESCAL-CL    |              |              |
|--------------------|--------|-------|-------------|-------|-------|--------------|--------------|--------------|
|                    |        |       | MRR         | H@1   | H@10  | MRR          | H@1          | H@10         |
| _similar_to        | 80     | 3     | 0.446       | 0.333 | 0.667 | <b>0.756</b> | <b>0.667</b> | <b>1.000</b> |
| _verb_group        | 1138   | 39    | 0.930       | 0.885 | 0.974 | 0.934        | <b>0.897</b> | 0.974        |
| *domain_usage      | 629    | 24    | 0.400       | 0.354 | 0.542 | <b>0.447</b> | <b>0.396</b> | 0.542        |
| *domain_region     | 923    | 26    | 0.329       | 0.269 | 0.442 | <b>0.360</b> | <b>0.289</b> | <b>0.500</b> |
| _member_meronym    | 7402   | 253   | 0.251       | 0.164 | 0.415 | <b>0.318</b> | <b>0.221</b> | <b>0.506</b> |
| _has_part          | 4816   | 172   | 0.223       | 0.151 | 0.375 | <b>0.245</b> | <b>0.174</b> | <b>0.384</b> |
| _hypernym          | 34796  | 1251  | 0.193       | 0.140 | 0.288 | <b>0.204</b> | <b>0.152</b> | <b>0.296</b> |
| _instance_hypernym | 2921   | 122   | 0.431       | 0.348 | 0.603 | <b>0.461</b> | <b>0.369</b> | <b>0.631</b> |
| *synset_domain*    | 3116   | 114   | 0.405       | 0.347 | 0.522 | <b>0.444</b> | <b>0.395</b> | <b>0.544</b> |
| *related_form      | 29715  | 1074  | 0.957       | 0.951 | 0.967 | 0.959        | 0.954        | 0.969        |
| _also_see          | 1299   | 56    | 0.606       | 0.554 | 0.679 | <b>0.621</b> | <b>0.571</b> | <b>0.696</b> |

Table 5: MRR, Hit@1 and Hit@10 results of RESCAL-DURA and RESCAL-CL methods on the triples with different relations in WN18RR dataset. We use \* to represent the abbreviation of some words in the relation names. #Train and #Test are the number of triples with the corresponding relations in the training and test set.

**Effect of Positive Instances** We apply an ablation study to verify the effect of positive instances. The variants of w/o Pos are the variants of RESCAL-CL and ComplEx-CL, which remove all original and additionally added positive instances in a mini-batch when calculating the contrastive loss. Table 6 shows the results of ablation study on WN18RR and FB15k-237 datasets. From Table 6 we know the positive instance of KGE can significantly improve the performance of KGE. Therefore, the positive instances we constructed are effective for the KGE.

**Sparsity Analysis** As mentioned by some other work (Zhang et al., 2020a), the sparsity (the number of zero entries) of embeddings can save the

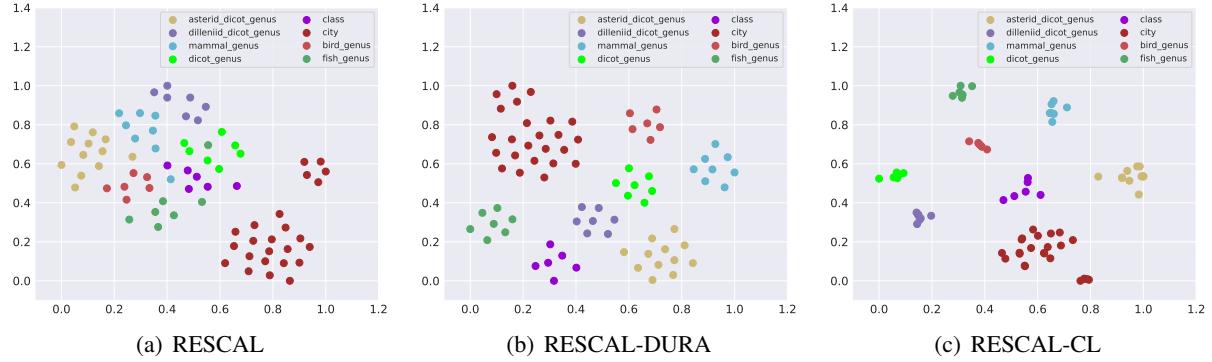


Figure 2: The visualization of the entity-relation couples’ embeddings using T-SNE. A points represents a  $(h_i, r_j)$  couple, and the points with the same color are the couples that connected with the same tail entity.

| Methods | Variants | WN18RR       |              | FB15k-237    |              |
|---------|----------|--------------|--------------|--------------|--------------|
|         |          | MRR          | Hit@10       | MRR          | Hit@10       |
| RESCAL  | w/o Pos  | 0.501        | 0.581        | 0.368        | 0.551        |
|         | w/ Pos   | <b>0.512</b> | <b>0.597</b> | <b>0.378</b> | <b>0.559</b> |
| ComplEx | w/o Pos  | 0.493        | 0.579        | 0.370        | 0.560        |
|         | w/ Pos   | <b>0.505</b> | <b>0.595</b> | <b>0.377</b> | <b>0.564</b> |

Table 6: Effect of Positive Instances.

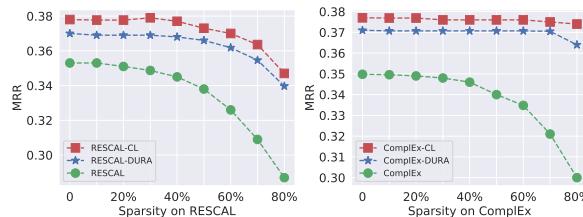


Figure 3: Effect of sparsity on FB15k-237.

storage usage of KG, which is vital for large scale real-world KGs. Therefore, we analyze the sparsity of embeddings training by our contrastive learning (CL) framework. We follow the setting in DURA (Zhang et al., 2020a), using a threshold  $\lambda$  to mask a proportion of entries as zero, and observe the MRR results under different proportions. Figure 3 shows the effect of embeddings’ sparsity on FB15k-237. We can find that the embeddings trained by CL have better results than DURA in the sparse version, so CL can better reduce the storage usage of KG and benefit the real-world KGs.

#### 5.4 Visualization

To make our method more explainable, we visualize the entity-relation couples via T-SNE (van der Maaten and Hinton, 2008). Specifically, we randomly pick up eight tail entities in WN18RR. We

find out the triples with these tail entities in the test set, and extract  $(h_i, r_j)$  couples in these triples. We visualize these couples’ embeddings trained by the RESCAL, RESCAL-DURA, and RESCAL-CL.

Figure 2 shows the results of visualization. The RESCAL method in Figure 2 (a) can not properly separate the couples with different tail entities. Compared with RESCAL, the RESCAL-DURA in Figure 2 (b) can relatively better separate the couples with different tail entities. However, since RESCAL-DURA can not capture the semantic similarity of couples with the same entity, the distribution of the couples connected with the same tail entity is still wide. Our RESCAL-CL can well split the couples in different types and shorten the distance of the couples connected with the same entity. Hence, our RESCAL-CL can better preserve the semantic information of the triples in knowledge graphs and has a higher performance.

## 6 Conclusion and Future Work

In this paper, we propose a simple yet efficient contrastive learning framework for TDB KGE to improve its performance. Compared with the previous work, our method can pull the related entities and entity-relation couples in different triples together in the semantic space and push the unrelated entities and couples apart. The experimental results on the standard datasets show that our method can achieve new state-of-the-art results. Our analyses further verify the effectiveness of our approach.

In the future, we plan to extend the critical insights of contrastive learning to distance based (DB) KGE methods and other representation learning problems in natural language processing.

## Acknowledgments

Zhiping Luo, Wentao Xu and Jian Yin are supported by the National Natural Science Foundation of China (U1811264, U1811262, U1811261, U1911203, U2001211), Guangdong Basic and Applied Basic Research Foundation (2019B1515130001), Key-Area Research and Development Program of Guangdong Province (2018B010107005, 2020B0101100001).

## References

- Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems 32*, pages 4463–4473.
- Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.
- Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.
- Xinlei Chen and Kaiming He. 2020. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*.
- Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818. AAAI Press.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.
- Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.
- Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189.
- Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113.
- Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.
- Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. 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 (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics.
- Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.
- Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*.
- Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.
- Timothee Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2863–2872. PMLR.
- Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, page 2181–2187. AAAI Press.
- Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research*.

- Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*.
- Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *arXiv preprint arXiv:2102.08473*.
- Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, volume 11, pages 809–816. PMLR.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.
- Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.
- Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality*.
- Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2071–2080. PMLR.
- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.
- Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844.
- Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, page 1112–1119. AAAI Press.
- Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742.
- Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466*.
- Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. TransG : A generative model for knowledge graph embedding. In *ACL*.
- Wentao Xu, Shun Zheng, Liang He, Bin Shao, Jian Yin, and Tie-Yan Liu. 2020. SEEK: Segmented embedding of knowledge graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3888–3897, Online. Association for Computational Linguistics.
- Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada. Association for Computational Linguistics.
- Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.
- Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems 32*, pages 2735–2745.
- Zhanqiu Zhang, Jianyu Cai, and Jie Wang. 2020a. Duality-induced regularizer for tensor factorization based knowledge graph completion. *Advances in Neural Information Processing Systems*, 33.
- Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020b. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press.