KGE-CL: Contrastive Learning of Knowledge Graph Embeddings

Wentao Xu,[1,4]* Zhiping Luo,[1,4]* Weiqing Liu, [2] Jiang Bian, [2] Jian Yin, [3,4] Tie-Yan Liu [2]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2]Microsoft Research Asia, Beijing, China
[3]School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China
[4]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
{xuwt6,luozhp7}@mail2.sysu.edu.cn
{weiqing.liu,jiang.bian,tyliu}@microsoft.com
issjyin@mail.sysu.edu.cn

Abstract

Learning the embeddings of knowledge graphs 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. However, most previous knowledge graph embedding 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 knowledge graph embeddings, which can shorten the semantic distance of the related entities and entity-relation couples in different triples and thus improve the expressiveness of knowledge graph embeddings. We evaluate our proposed method on three standard knowledge graph benchmarks. It is noteworthy that our method can yield some new state-of-the-art results, achieving 51.2% MRR, 46.8% Hits@1 on the WN18RR dataset, and 59.1% MRR, 51.8% Hits@1 on the YAGO3-10 dataset.

1 Introduction

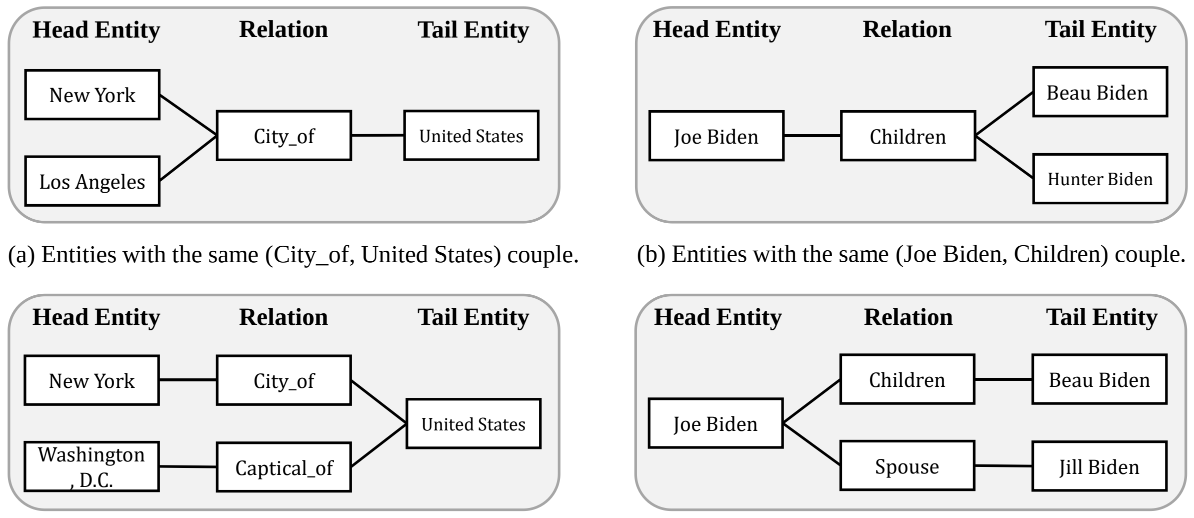
The knowledge graph (KG) stores a vast number of human knowledge in the format of triples. A triple (h, r, t) in a knowledge graph contains a head entity h, a tail entity t, and a relation r between h and t. The knowledge graph embedding (KGE) aims to project the massive interconnected entities and relations in a knowledge graph into vectors or matrices, which can preserve the semantic information of the triples. Learning the embeddings of knowledge graph 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, Salakhutdinov, and Gupta 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 knowledge graph embeddings is maximizing the score of a true triple (h_i, r_j, t_k) and minimizing the score of a false tripe (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, Huang, and Zhu 2016). The other category is the tensor decomposition based (TDB) methods, which treat a knowledge graph 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, Tresp, and Kriegel 2011) and ComplEx (Trouillon et al. 2016).

However, the previous methods overlook the connections between the related entities and entity-relation couples in different triples. They separately feed each triple into the scoring function and optimize the scoring function' output to learn the entities' and relations' embeddings. In a knowledge graph, some entities (entity-relation couples) that share the same entity-relation couple (entity) have similar semantics. Capturing the semantic similarity of these entities or couples can improve the expressiveness of embeddings, which indicates the performance in capturing semantic information (Dettmers et al. 2018; Xu et al. 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*, 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 knowledge graph embeddings, which is quite general for existing KGE 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

* The first two authors contributed equally to this work.



(a) Entities with the same (City_of, United States) couple.

(b) Entities with the same (Joe Biden, Children) couple.

(c) Couples with the same tail entity United States.

(d) Couples with the same head entity Joe Biden.

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

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, Biega, and Suchanek 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, and 59.1% MRR, 51.8% Hits@1 on the YAGO3-10 dataset. Moreover, We further analyze the improvement by comparing our method with the baseline on the triples with different relations. At last, to clearly explain why our method outperforms existing methods, we conduct the visualization of the knowledge graph embeddings 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 knowledge graph embeddings. It can capture the semantic similarity of the related entities and entity-relation couples in different triples, thus improving the expressiveness of embeddings. To our best knowledge, the KGE-CL is the first contrastive learning framework of knowledge graph embeddings.
- 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 methods and tensor decomposition based KGE.

Distance based (DB) methods describe relations as relational maps between head and tail entities. They usually use the Minkowski distance to score a given triplet. 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, Huang, and Zhu 2016). The structured embedding (SE) (Bordes et al. 2011) utilizes the relations as linear maps on the head and tail entities, and its scoring function is: $f(h_i, r_j, t_k) = -\|\mathbf{R}_j^1\mathbf{h}_i - \mathbf{R}_j^2\mathbf{t}_k\|_1$. The RotatE (Sun et al. 2019) uses each relation as a rotation in a complex vector space and its scoring function is calculated by: $f(h_i, r_j, t_k) = -\|\mathbf{h}_i \circ \mathbf{r}_j - \mathbf{t}_k\|_1$, where $\mathbf{h}_i, \mathbf{r}_j, \mathbf{t}_k \in \mathbb{C}^k$ and $|[\mathbf{r}]_i| = 1$.

Tensor decomposition based (TDB) KGE methods formulate the KGE task as a third-order binary tensor decomposition problem. RESCAL (Nickel, Tresp, and Kriegel 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, Usunier, and Obozinski 2018) and DURA (Zhang, Cai, and Wang 2020) regularizers. These regularizers bring more significant improvements than the original squared Frobenius norm (L_2 norm) regularizer (Nickel, Tresp, and Kriegel 2011; Yang et al. 2015; Trouillon et al. 2016).

However, both of the existing distance based and tensor decomposition based methods separately optimize each triple with the scoring function, overlooking the relationships between the entities or entity-relation couples in different triples. Therefore, they can not correctly capture the semantic similarity between these related entities and couples in different triples. To address the limitations of existing work, we propose our KGE-CL method for learning the knowledge graph embeddings.

3 Backgrounds

In this section, we will introduce the background of knowledge graph embedding in Setcion 3.1, and the background of contrastive learning in Section 3.2.

3.1 Knowledge Graph Embedding

Knowledge Graph Given a entities set $\mathcal{E}$ and a relations set $\mathcal{R}$, a knowledge graph $\mathcal{K} = \{(h_i, r_j, t_k)\}$ is a set of triples, where $h_i \in \mathcal{E}$ is the head entity, $r_j \in \mathcal{R}$ is the relation, and $t_k \in \mathcal{E}$ is the tail entity.

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. Exiting KGE work contains two important categories: distance based methods and tensor decomposition based methods.

Distance Based (DB) KGE Methods DB methods define the scoring function $f(h_i, r_j, t_k)$ with the Minkowski distance (Bordes et al. 2013; Wang et al. 2014; Lin et al. 2015; Ji et al. 2015; Xiao, Huang, and Zhu 2016), and the scoring functions of this kind of methods is: $f(h_i, r_j, t_k) = -\|\Gamma(h_i, r_j, t_k)\|_p$, where Γ is a model-specific function.

Tensor Decomposition Based (TDB) KGE Methods TDB methods like RESCAL (Nickel, Tresp, and Kriegel 2011) and ComplEx (Trouillon et al. 2016), regard a knowledge graph 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 $\mathcal{X}_j \approx \mathbf{Re}\left(\mathbf{H}\mathbf{R}_j\overline{\mathbf{T}}^\top\right)$, 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, $\mathbf{Re}\left(\cdot\right)$ and $\bar{\cdot}$ are the real part and the conjugate of a complex matrix, respectively. Then the scoring functions of TDB KGE methods is: $f(h_i, r_j, t_k) = \mathbf{Re}\left(\mathbf{h}_i\mathbf{R}_j\bar{\mathbf{t}}_k^\top\right)$. Note that the real part and the conjugate of a real matrix are itself. The goal of TDB models is to seek matrices $\mathbf{H}, \mathbf{R}_1, \ldots, \mathbf{R}_{|\mathcal{R}|}, \mathbf{T}$, such that $\mathbf{Re}\left(\mathbf{H}\mathbf{R}_j\overline{\mathbf{T}}^\top\right)$ 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, Chopra, and LeCun 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^+:

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

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. 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 easily apply to arbitrary tensor decomposition based 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 knowledge graph embeddings, then we introduce the training objective of our method.

4.1 Contrastive Loss of Knowledge Graph Embeddings

In this subsection, we elaborate on the contrastive loss that we designed for knowledge graph embeddings.

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 knowledge graph embeddings. To capture the interactions between triples in a knowledge graph, we design a new approach to construct the positive instances for knowledge graph embeddings. For a triple (h_i, r_j, t_k), the corresponding scoring function of TDB methods is:

$$f(h_i, r_j, t_k) = \mathbf{Re}\left(\mathbf{h}_i\mathbf{R}_j\bar{\mathbf{t}}_k^\top\right) = \mathbf{Re}\left(\langle \mathbf{h}_i\mathbf{R}_j, \bar{\mathbf{t}}_k\rangle\right) = \mathbf{Re}\left(\langle \mathbf{h}_i, \mathbf{R}_j\bar{\mathbf{t}}_k\rangle\right). \tag{2}$$

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 $\mathbf{R}_j\bar{\mathbf{t}}_k$ 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 $\mathbf{R}_j\bar{\mathbf{t}}_k$ 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 $(\mathbf{R}_j\bar{\mathbf{t}}_k)^+$ 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. For an instance h_i, we will use all of its positive instances h_i^+ in the same mini-batch.

Base Encoder Motivated by the existing contrastive learning frameworks, we utilize a base encoder $g(\cdot)$ to encode the original and positive instances in the four types of positive pairs: (h_i, h_i^+), (t_k, t_k^+), $((h_i, r_j), (h_i, r_j)^+)$ and $((r_j, t_k), (r_j, t_k)^+)$. The base encoder $g(\cdot)$ we used is a 2-layers MLP, which has a batch normalization (Ioffe and Szegedy 2015) and a ReLU activation function in each hidden layer. To make the notation more convenient and clear, we use the same notation to represent the output of encoder, e.g., $\mathbf{h}_i = g(\mathbf{h}_i)$, $(\mathbf{h}_i\mathbf{R}_j)^+ = g((\mathbf{h}_i\mathbf{R}_j)^+)$.

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 after feeding the instances into the base encoder, and the overall contrastive loss for a triple (h_i, r_j, t_k) is:

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

Discussion We analyze Equation 3 from the aspect of gradient. Taking the contrastive loss term $\mathrm{CL}(\mathbf{h}_i, \mathbf{h}_i^+)$ as an example, the gradient of $\mathrm{CL}(\mathbf{h}_i, \mathbf{h}_i^+)$ to embedding $\mathbf{h}_i$ is:

$$\begin{aligned}&\frac{\partial \mathrm{CL}(\mathbf{h}_i, \mathbf{h}_i^+)}{\partial \mathbf{h}_i} \\ &= \frac{-1}{|P(i)|}\frac{\partial}{\partial \mathbf{h}_i}\left(\sum_{h_i^+\in P(i)} \frac{\mathbf{h}_i \cdot \mathbf{h}_i^+}{\tau} - \log\sum_{h_j\in N(i)} e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}\right) \\ &= \frac{-1}{\tau|P(i)|}\left(\sum_{h_i^+\in P(i)} \mathbf{h}_i^+ - \frac{\sum_{h_j\in N(i)}\left(e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}\mathbf{h}_j\right)}{\sum_{h_j\in N(i)} e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}}\right).\end{aligned} \tag{4}$$

Then when we update the embedding $\mathbf{h}_i$ with the gradient:

$$\begin{aligned}\mathbf{h}_i^{t+1} &= \mathbf{h}_i^t - \eta\frac{\partial \mathrm{CL}(\mathbf{h}_i, \mathbf{h}_i^+)}{\partial \mathbf{h}_i} \\ &= \mathbf{h}_i^t + \frac{\eta\sum_{h_i^+\in P(i)}\mathbf{h}_i^+}{\tau|P(i)|} - \frac{\eta\sum_{h_j\in N(i)}\left(e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}\mathbf{h}_j\right)}{\tau|P(i)|\sum_{h_j\in N(i)} e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}}\end{aligned} \tag{5}$$

The Equation 5 shows 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 $\frac{\eta\sum_{h_j\in N(i)}\left(e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}\mathbf{h}_j\right)}{\tau|P(i)|\sum_{h_j\in N(i)} e^{(\mathbf{h}_i\cdot\mathbf{h}_j/\tau)}}$ 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 $\mathrm{CL}(\mathbf{h}_i\mathbf{R}_j, (\mathbf{h}_i\mathbf{R}_j)^+)$ achieves the highest results, while in FB15k-237 and YAGO3-10 datasets, $\mathrm{CL}(\mathbf{R}_j\bar{\mathbf{t}}_k, (\mathbf{R}_j\bar{\mathbf{t}}_k)^+)$ is the best. Hence, we introduce a weighted contrastive loss, assigning a weight α_* for each contrastive loss term, and α_* is a hyper-parameter that can be flexibly tuned for a specific knowledge graph. The contrastive loss $\mathcal{L}_c^w(h_i, r_j, t_k)$ after adding weights α_* is:

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

Contrastive Loss	**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
$\mathrm{CL}(\mathbf{h}_i, \mathbf{h}_i^+)$	.509	.465	**.527**	.588	.362	.269	.397	.545	.575	.501	.617	.712
$\mathrm{CL}(\mathbf{t}_k, \mathbf{t}_k^+)$	.509	.468	.425	.587	.361	.269	.398	.544	.574	.497	.618	.712
$\mathrm{CL}(\mathbf{h}_i\mathbf{R}_j, (\mathbf{h}_i\mathbf{R}_j)^+)$	**.512**	**.469**	**.527**	**.595**	.357	.265	.392	.540	.559	.479	.604	.707
$\mathrm{CL}(\mathbf{R}_j\bar{\mathbf{t}}_k, (\mathbf{R}_j\bar{\mathbf{t}}_k)^+)$	.504	.462	.516	.580	**.370**	**.278**	**.409**	**.552**	**.581**	**.507**	**.625**	**.713**

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

4.2 Training Objective

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

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

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.

$\mathcal{L}_s$ **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, Bajgar, and Kleindienst 2017) to calculate the distance between the scoring function's output and the triple's label. Since (Lacroix, Usunier, and Obozinski 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, Tresp, and Kriegel 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 knowledge graph embeddings, such as the N3 (Lacroix, Usunier, and Obozinski 2018) and DURA (Zhang, Cai, and Wang 2020) regularizers. Since the (Zhang, Cai, and Wang 2020) 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. 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 knowledge graph datasets—WN18RR (Toutanova and Chen 2015), FB15k-237 (Dettmers et al. 2018), and YAGO3-10 (Mahdisoltani, Biega, and Suchanek 2015) to evaluate the performance of knowledge graph embeddings. We divide the datasets into training, validation, 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 (Mahdisoltani, Biega, and Suchanek 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.

	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.

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, Hazan, and Singer 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 number of encoder's hidden units m in {512, 1024, 2048, 4096} and the temperature τ in Equation 1 in {0.3, 0.5, 0.7, 0.9, 1.0}. We search the weights α_h, α_t, α_{hr} and α_{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 4. Besides, the batch size is 512 for RESCAL-CL and 200 for ComplEx-CL, and the learning rate η 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.

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
RESCAL-CL†	**.512**	**.468**	**.531**	**.597**	.370	.278	.406	.554	.581	.507	.625	.713
ComplEx-CL†	.505	.458	.522	.595	**.373**	**.279**	**.410**	**.564**	**.591**	**.518**	**.634**	**.722**

† Statistically significant improvements by independent t-test with $p = 0.01$.

Table 3: 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, Cai, and Wang 2020), and the results of other baselines are from their original papers.

Datasets	**Methods**	d	m	τ	α_h	α_t	α_{hr}	α_{tr}
WN18RR	RESCAL-CL	512	512	0.9	0	0	2.0	0
	ComplEx-CL	2000	2048	0.5	0	0	0	2.0
FB15k-237	RESCAL-CL	512	512	0.9	0	0	0	2.0
	ComplEx-CL	2000	2048	0.5	2.0	0	0	0
YAGO3-10	RESCAL-CL	512	512	0.9	0	0	0	1.0
	ComplEx-CL	2000	2048	0.5	0	1.0	0	0

Table 4: The selection of the hyper-parameters of RESCAL-CL and ComplEx-CL on different datasets.

Compared Methods We compare our KGE-CL method with existing state-of-the-art knowledge graph embedding methods, including CP (Hitchcock 1927), RESCAL (Nickel, Tresp, and Kriegel 2011), ComplEx (Trouillon et al. 2016), ConvE (Dettmers et al. 2018), RoratE (Sun et al. 2019), MuRP (Balazevic, Allen, and Hospedales 2019), HAKE (Zhang et al. 2020), ComplEx-N3 (Lacroix, Usunier, and Obozinski 2018), ROTH (Chami et al. 2020), REFE (Chami et al. 2020), CP-DURA (Zhang, Cai, and Wang 2020), RESCAL-DURA (Zhang, Cai, and Wang 2020) and ComplEx-DURA (Zhang, Cai, and Wang 2020).

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 knowledge graph embeddings. 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 N in prediction.

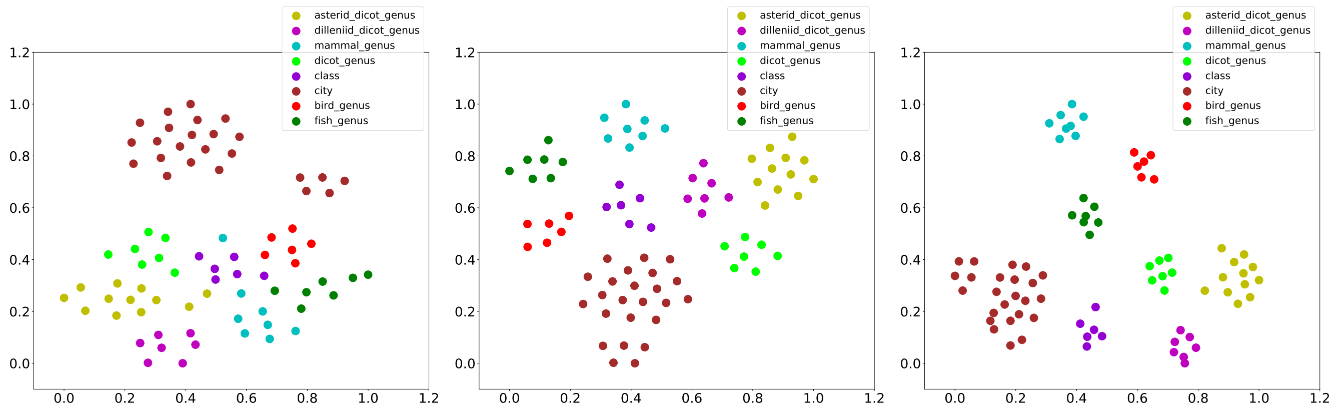
For both metrics, we remove some of the corrupted triples that exist in datasets from the ranking results, which is also called *filtered* setting in (Bordes et al. 2013). For the metrics of Hits@N, we use Hits@1, Hits@3, and Hits@10.

Table 3 shows the results of link prediction on WN18RR, FB15K-237, and YAGO3-10 datasets. Our proposed method achieves the highest results 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 improvements of RESCAL-CL and ComplEx-CL are relatively not that obvious, which implies that capturing the semantic similarity between the related entities and entity-relation couples is not that important in FB15k-237 dataset.

5.3 Model Analysis

Analyzing the Improvements To further explore why our method outperforms existing state-of-the-art techniques, we compare our proposed 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 in a specific relation.

Effect of Positive Instances We apply an ablation study to verify the effect of positive instances. The RESCAL-



(a) RESCAL (b) RESCAL-DURA (c) RESCAL-CL

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.

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	**0.756**	**0.667**	**1.000**
_verb_group	1138	39	0.930	0.885	0.974	0.934	**0.897**	0.974
*domain_usage	629	24	0.400	0.354	0.542	**0.447**	**0.396**	0.542
*domain_region	923	26	0.329	0.269	0.442	**0.360**	**0.289**	**0.500**
_member_meronym	7402	253	0.251	0.164	0.415	**0.318**	**0.221**	**0.506**
_has_part	4816	172	0.223	0.151	0.375	**0.245**	**0.174**	**0.384**
_hypernym	34796	1251	0.193	0.140	0.288	**0.204**	**0.152**	**0.296**
_instance_hypernym	2921	122	0.431	0.348	0.603	**0.461**	**0.369**	**0.631**
_synset_domain*	3116	114	0.405	0.347	0.522	**0.444**	**0.395**	**0.544**
*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	**0.621**	**0.571**	**0.696**

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 set and test set.

CL_nopos and Complex-CL_nopos are the variants of RESCAL-CL and Complex-CL, which remove all positive instances in a mini-batch when calculating the contrastive loss. Table 6 shows the results of RESCAL-CL_nopos, RESCAL-CL, Complex-CL_nopos, and Complex-CL on WN18RR and FB15k-237 datasets. From Table 6 we know the positive instance of knowledge graph embedding can significantly improve the performance of knowledge graph embeddings. Therefore, the positive instances we constructed are effective for the knowledge graph embeddings.

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. We extract the (h_i, r_j) couples in these triples and 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 expressiveness.

Methods	WN18RR		FB15k-237	
	MRR	Hit@10	MRR	Hit@10
RESCAL-CL_nopos	0.501	0.581	0.368	0.551
RESCAL-CL	**0.512**	**0.597**	**0.370**	**0.554**
ComplEx-CL_nopos	0.493	0.579	0.370	0.560
ComplEx-CL	**0.505**	**0.595**	**0.373**	**0.564**

Table 6: Effect of Positive Instances

6 Conclusion and Future Work

In this paper, we propose a simple yet efficient contrastive learning framework for knowledge graph embeddings to improve its expressiveness. 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 other representation learning problems in natural language processing.

References

Balazevic, I.; Allen, C.; and Hospedales, T. 2019. Multi-relational Poincaré Graph Embeddings. In *Advances in Neural Information Processing Systems 32*, 4463–4473.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, 2787–2795. Curran Associates, Inc.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 301–306. AAAI Press.

Chami, I.; Wolf, A.; Juan, D.-C.; Sala, F.; Ravi, S.; and Ré, C. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6901–6914. Online: Association for Computational Linguistics.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, X.; and He, K. 2020. Exploring Simple Siamese Representation Learning. *arXiv preprint arXiv:2011.10566*.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, 1811–1818. AAAI Press.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159.

Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, 1735–1742. IEEE.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9729–9738.

Hitchcock, F. L. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4): 164–189.

Huang, X.; Zhang, J.; Li, D.; and Li, P. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 105–113.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 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)*, 687–696. Beijing, China: Association for Computational Linguistics.

Ji, G.; Liu, K.; He, S.; and Zhao, J. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Kadlec, R.; Bajgar, O.; and Kleindienst, J. 2017. Knowledge base completion: Baselines strike back. *arXiv preprint arXiv:1705.10744*.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Lacroix, T.; Usunier, N.; and Obozinski, G. 2018. Canonical Tensor Decomposition for Knowledge Base Completion. In *Proceedings of the 35th International Conference on Machine Learning*, 2863–2872. PMLR.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2181–2187. AAAI Press.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. M. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research*.

Marino, K.; Salakhutdinov, R.; and Gupta, A. 2016. The more you know: Using knowledge graphs for image classification. *arXiv preprint arXiv:1612.04844*.

Meng, Y.; Xiong, C.; Bajaj, P.; Tiwary, S.; Bennett, P.; Han, J.; and Song, X. 2021. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *arXiv preprint arXiv:2102.08473*.

Nickel, M.; Tresp, V.; and Kriegel, H.-P. 2011. A Three-way Model for Collective Learning on Multi-relational Data. In *Proceedings of the 28th International Conference on Machine Learning*, volume 11, 809–816. PMLR.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

Toutanova, K.; and Chen, D. 2015. Observed Versus Latent Features for Knowledge Base and Text Inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, 2071–2080. PMLR.

van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605.

Wang, H.; Zhang, F.; Xie, X.; and Guo, M. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, 1835–1844.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1112–1119. AAAI Press.

Wu, Z.; Wang, S.; Gu, J.; Khabsa, M.; Sun, F.; and Ma, H. 2020. CLEAR: Contrastive Learning for Sentence Representation. *arXiv preprint arXiv:2012.15466*.

Wu, Z.; Xiong, Y.; Yu, S. X.; and Lin, D. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3733–3742.

Xiao, H.; Huang, M.; and Zhu, X. 2016. TransG : A Generative Model for Knowledge Graph Embedding. In *ACL*.

Xu, W.; Zheng, S.; He, L.; Shao, B.; Yin, J.; and Liu, T.-Y. 2020. SEEK: Segmented Embedding of Knowledge Graphs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3888–3897. Online: Association for Computational Linguistics.

Yang, B.; and Mitchell, T. 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)*, 1436–1446. Vancouver, Canada: Association for Computational Linguistics.

Yang, B.; Yih, S. W.-t.; He, X.; Gao, J.; and Deng, L. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q. 2019. Quaternion Knowledge Graph Embeddings. In *Advances in Neural Information Processing Systems 32*, 2735–2745.

Zhang, Z.; Cai, J.; and Wang, J. 2020. Duality-Induced Regularizer for Tensor Factorization Based Knowledge Graph Completion. *Advances in Neural Information Processing Systems*, 33.

Zhang, Z.; Cai, J.; Zhang, Y.; and Wang, J. 2020. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press.