Self-Knowledge Distillation for Knowledge Graph Embedding

Haotian Xu[1], Yuhua Wang[2], Jiahui Fan[3]
[1]University of Science and Technology of China, Hefei, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]Meituan Group, Beijing, China
xuhaotian@mail.ustc.edu.cn, wangyuhua2021@ia.ac.cn, fanjiahui06@meituan.com

Abstract

Knowledge graph embedding (KGE) is an important task and it can benefit lots of downstream applications. General KGE can increase the embedding dimension to improve model performance. High-dimensional KGE will significantly increase the number of model parameters and training time. Therefore, knowledge distillation is proposed for learning a low-dimensional model from a pre-trained high-dimensional model. To avoid introducing a complex teacher model, we use self-knowledge distillation. However, there are still some issues with the self-knowledge distillation model we mentioned later. One of them is misdirection from incorrect predictions during model training. Another is the loss of discrimination information caused by excessive distillation temperature. To address these issues, we apply self-knowledge distillation, knowledge adjustment and dynamic temperature distillation to KGE. Self-knowledge distillation uses the information from the latest iteration to guide the training in the current iteration. Knowledge adjustment fixes the predictions of misjudged training samples. Dynamic temperature distillation designs dynamic sample-wise temperatures to compute soft targets. Our model can not only improve model performance but also achieve a lightweight model. Experimental results demonstrate the effectiveness and generalization ability of our model in link prediction. The lightweight model can maintain good model performance while reducing the number of model parameters and training time.

Keywords: knowledge graph embedding, self-knowledge distillation, link prediction

1. Introduction

Knowledge Graph (KG) is an important branch of technology of artificial intelligence and a structured semantic knowledge base. KG is used to describe concepts and their interrelationships in the physical world in the form of symbols (Ji et al., 2021). The basic unit of KG is the "entity-relation-entity" triple. Entities are connected through relations to form a networked knowledge structure. KGs are generally constructed manually or semi-automatically. Most of the KGs are incomplete or sparse, and many hidden relations have not been discovered. Knowledge graph embedding (KGE) (Bordes et al., 2013) is proposed to address these issues. The key idea is to learn the embedding of entities and relations while preserving the original structure of the KG. The embedding of entities and relations can be further applied to various downstream tasks.

The general KGE method will obtain better performance by increasing the dimension of embedding (Zhu et al., 2022). Not only the number of model parameters but also the cost of training time will greatly increase with the rise of the embedding dimension. Therefore, knowledge distillation (Hinton et al., 2015) is proposed to apply to KGE. The goal is to learn low-dimensional KGE from a pre-trained high-dimensional one (Zhu et al., 2022).

Early knowledge distillation methods (Hinton et al., 2015; Romero et al., 2015) employ offline distillation, which transfers the knowledge of a pre-trained teacher model to the student model. To alleviate the large amount of training time brought by the teacher model in offline distillation, online distillation is proposed (Zhang et al., 2018). The key idea is that the parameters of the teacher model and the student model will be updated at the same time. However, existing offline and online distillation methods usually cannot avoid the problem of computational sources and run-in memory consumed by the complex teacher model. To avoid the above problems, the idea of self-knowledge distillation (Hahn and Choi, 2019) is proposed. Finding a teacher model without adding a large model can also provide an effective gain of information to the student model. The teacher model here is often not more complicated than the student model, and may even be the student model itself.

Inspired by the success of self-knowledge distillation (Yuan et al., 2020; Kim et al., 2021; Shen et al., 2022) in computer vision, we plan to apply self-knowledge distillation to KGE. Although self-knowledge distillation does not need to introduce a teacher model, there are some other issues with self-knowledge distillation. Firstly, because the teacher model in self-knowledge distillation is itself, the accuracy of the predictions in the early stage of model training is not high. This will lead to more incorrect predictions early in the model training. To fix the misleading model training caused by incorrect predictions, knowledge adjustment (KA) (Wen et al., 2021) is adopted. For KA, the predictions

LREC-COLING 2024, pages 14595–14605

of misjudged training samples are fixed according to the corresponding ground truth before loss function calculation. The student model trained in this way is fed completely correct information from the teacher model. Secondly, previous works (Hinton et al., 2015) in knowledge distillation indicate that students can benefit from the uncertainty of supervision, and overly uncertain predictions of teachers may also affect the student. For example, when the distillation temperature is high, some discrimination information is lost. To solve this problem, dynamic temperature distillation (DTD) (Wen et al., 2021) is used. For DTD, we detect that some uncertainty of soft targets comes from excessive temperature, thus we turn to design dynamic sample-wise temperature to calculate soft targets. By conducting so, student training will gain more discrimination information on confusing samples.

In this paper, we apply self-knowledge distillation, KA and DTD to KGE. Self-knowledge distillation uses the distillation from the latest batch to generate soft targets to guide the training in the current batch. The introduction of self-knowledge distillation can not only reduce the model size to achieve lightweight but also avoid the consumption of computational sources and run-in memory caused by the introduction of a complex teacher model. Then, we use DTD to design dynamic sample-wise temperatures to compute soft targets. KA is used to fix the predictions of misjudged training samples. We propose a simple but effective method to apply Self-knowledge distillation, Knowledge adjustment and Dynamic temperature distillation to knowledge graph Embedding (SKDE). For SKDE, the key idea is that the target network plays both the roles of teacher and student in each batch during the training process. As a teacher, the target network provides soft targets to guide itself in the next iteration. As a student, the target network distills soft targets generated from the latest iteration and minimizes the loss function with ground truth. Our goal is to obtain a lightweight model while maintaining a good model performance.

We validate the comprehensive effectiveness of our methods on two standard benchmark datasets, namely WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015). We select four representative KGE methods to illustrate the generalization ability of our methods, including DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018) and AcrE (Ren et al., 2020).

The major contributions are three-fold:

- Based on self-knowledge distillation, KA and DTD, an effective and uncomplicated KGE method is proposed. Our SKDE simply utilizes the data updated in the last iteration to guide the training in the current iteration. Without changing the network structure of KGE, the implementation of our SKDE only needs to increase a small number of computational sources and run-in memory, and the number of model parameters will not increase. To the best of our knowledge, we are the first to apply self-knowledge distillation to KGE.

- Aiming at the shortcomings of self-knowledge distillation, we put forward an improvement scheme. We use DTD to design dynamic sample-wise temperatures to compute soft targets. Then, we use KA to fix the predictions of misjudged training samples. The experimental results show that both KA and DTD can effectively optimize the model performance.

- Extensive experiments demonstrate the effectiveness and generalization ability of our SKDE, which can be effectively applied to most KGEs. Compared with other KGE methods, our SKDE can obtain better performance without increasing the number of model parameters. SKDE can still maintain a good performance after reducing the number of model parameters. At the same time, the training time of the model also decreases as the number of model parameters decreases. In this way, we not only gain a lightweight model but also maintain a good model performance.

2. Related Works

To make this paper self-contained, we introduce some related topics here on knowledge graph embedding and knowledge distillation.

2.1. Knowledge Graph Embedding

In recent years, KGE technology has developed rapidly and has been widely used. The key idea of KGE is to transform the entities and relations in the KG into a continuous vector space as vector representations (Zhu et al., 2022). Subsequently, these vectors can be applied to many downstream tasks of KG. This article mainly involves the following types of KGE methods (Zhao et al., 2021): (i) Translational-based models, which view relations as translations from head entity to tail entity, such as TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016). (ii) Tensor factorization-based models, which suppose the score of a triple can be factorized into several tensors, such as RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015). (iii) CNN-based models, which use convolutions on entities and relations embedding to link prediction, such as ConvE (Dettmers et al., 2018), AcrE (Ren et al., 2020).

Although there are so many types of KGE, a common problem is that high-dimensional KGE will bring huge challenges to computational sources and run-in memory. While the model is lightweight, it is also necessary to keep the performance of the model. At present, the research on lightweight KGE mainly uses knowledge distillation. MulDE (Wang et al., 2021) is the first work to apply knowledge distillation to KGE. MulDE transfers the knowledge from multiple teachers to a student. DualDE (Zhu et al., 2022) also uses knowledge distillation to KGE. DualDE considers the dual-influence between the teacher and the student. The above methods inevitably introduce complex teacher models. In this work, we propose a novel KGE method based on self-knowledge distillation. Our method can achieve a lightweight model without requiring a complex teacher model.

2.2. Knowledge Distillation

Knowledge distillation, first introduced by Buciluǎ et al. (2006), was later popularized by Hinton et al. (2015). Numerous studies have shown that knowledge distillation is effective in many places. The goal of knowledge distillation is to transfer knowledge from a high-capacity teacher model to a lightweight student model. Although offline distillation (Hinton et al., 2015; Romero et al., 2015) has a good performance, pre-training a complex teacher model requires a lot of training time and computational sources. To alleviate the problem of massive time consumption, online distillation is proposed (Zhang et al., 2018). Online distillation can achieve similar performance to offline distillation while significantly improving computational efficiency. However, the optimization of online distillation involves multiple networks, which requires extra run-in memory to store all parameters.

To improve the efficiency and effectiveness of knowledge transfer, self-knowledge distillation (Hahn and Choi, 2019) is proposed. Self-knowledge distillation utilizes knowledge from itself and avoids introducing additional networks. In teacher-free knowledge distillation (Tf-KD) (Yuan et al., 2020), the student model learns from itself or manually designed regularization distribution. The author of the Tf-KD finds through experiments and analyses that the "dark knowledge" of a teacher model is more of a regularization term than similarity information of categories. In progressive self-knowledge distillation (PS-KD) (Kim et al., 2021), the model progressively distills its own knowledge to soften hard targets during training. Targets are adjusted adaptively by combining the ground truth and past predictions from the model itself. Self-Distillation from Last Mini-Batch (DLB) (Shen et al., 2022) rearranges the sequential sampling by constraining half of each mini-batch coinciding with the previous iteration. The rest half will coincide with the upcoming iteration. The former half mini-batch distills on-the-fly soft targets generated in the previous iteration. In self-distillation framework with meta learning (MetaSD) (Li et al., 2023), the source and pruned models co-evolve in a dynamic manner during training, thus we can avoid pre-training a large model in advance, and the performance of the pruned model is not limited to that of a pre-trained large model.

However, there are still some issues in the existing self-knowledge distillation model. One of them is misdirection from incorrect predictions during training. Another is the loss of discrimination information caused by excessive distillation temperature. To address these issues in the above self-knowledge distillation model, we apply KA and DTD to self-knowledge distillation. Our model obtains more discrimination information through DTD and fixes incorrect predictions through KA.

3. Methods

We begin this section by introducing the notations and definitions used in the rest of the paper, followed by a background on loss functions. Then, we introduce the various modules in SKDE. These include self-knowledge distillation, dynamic temperature distillation and knowledge adjustment. The overall training process for our SKDE is visualized in Fig. 1.

3.1. Background

KGs are denoted by $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ represent the set of entities and relations, and $\mathcal{T}$ denotes the triples (facts) of the form $\{(h, r, t)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. A triple (h, r, t) is represented as a relation r between head entity h and tail entity t in $\mathcal{G}$. KGE tries to learn an effective representation of entities, relations, and a scoring function f, such that for a given input triple $x = (h, r, t)$, $f(x)$ gives the likelihood of triple x being a valid triple (Nathani et al., 2019).

We write a K-classes labelled dataset as $\mathcal{D} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, where N is the total number of training triples and K is the total number of entities in $\mathcal{E}$. In every batch, a batch of n samples $\mathcal{B} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{n} \subseteq \mathcal{D}$ is provided to the model, where $x_i = (h_i, r_i, t_i)$ is a valid triple, $\mathbf{y}_i \in \mathbb{R}^{1\times K}$ is a label vector whose elements are ones for entities that exist in the triple x_i and zero otherwise.

Most CNN-based KGEs (like ConvE (Dettmers et al., 2018), AcrE (Ren et al., 2020), etc.) use a kind of softmax regression loss that considers the ranking scores collectively (listwise ranking method). In our SKDE, we define the same listwise loss function as used in ConvE and AcrE.

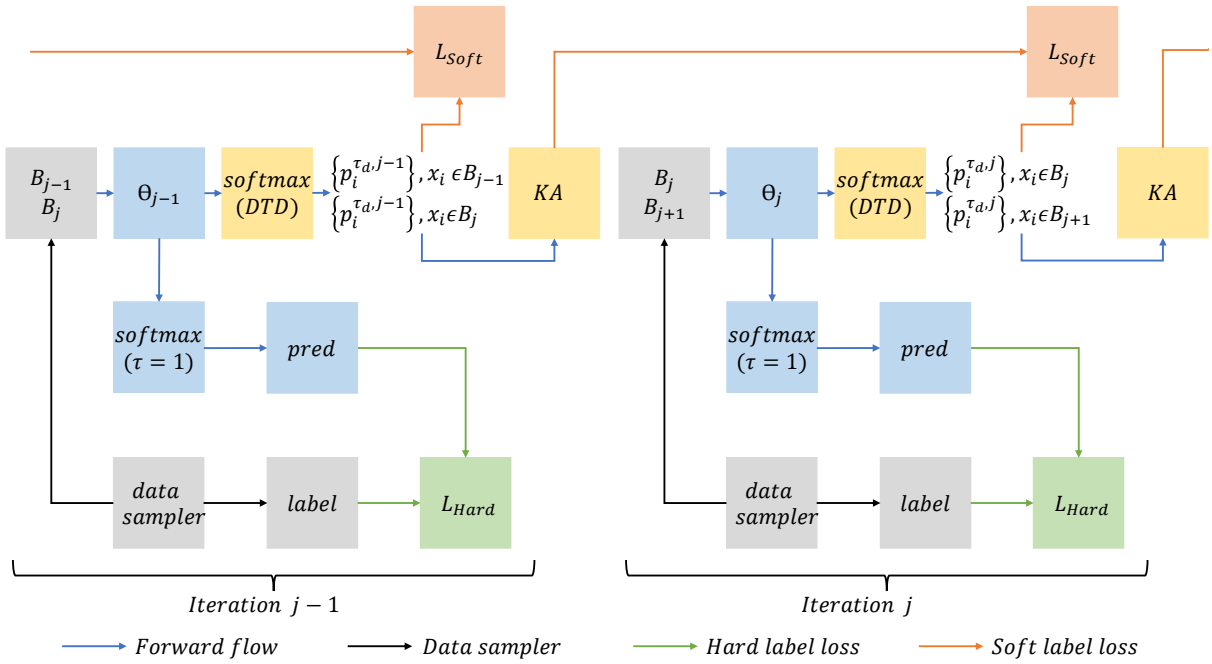

Figure 1: The overall training process for our proposed SKDE. We denote $\mathcal{B}_j$, θ_j and $\mathbf{p}_i^{\tau,j}$ for a batch of data samples, network parameters and soft target probability in the j^{th} iteration. In this approach, a portion of the model's output is used to calculate the hard label loss, while another portion is used to obtain predicted results through DTD. The predicted results from $\mathcal{B}_{j-1}$ are used to calculate the soft label loss, while the predicted results from $\mathcal{B}_j$ are utilized with KA to guide training in the next iteration.

$$\mathcal{L}_{BCE}(\mathbf{y}_i, \mathbf{p}_i^\tau) = -\frac{1}{K}\sum_{k=1}^{K}(y_i(k)\cdot\log p_i^\tau(k) + (1-y_i(k))\cdot\log(1-p_i^\tau(k))) \quad (1)$$

where temperature τ is usually set to 1 in (1). This loss function takes one (h_i, r_i) pair and scores it against all entities simultaneously. Thus, the model is very fast for both training and inference (Ren et al., 2020). In order to predict tail entity of (h_i, r_i), the predictive distribution $\mathbf{p}_i^\tau = (p_i^\tau(1), ..., p_i^\tau(K)) \in \mathbb{R}^{1\times K}$ in a softmax classifier for class $k \in [K]$ is formulated as:

$$p_i^\tau(k) = \frac{\exp(f(h_i, r_i, t_k)/\tau)}{\sum_{j=1}^{K}\exp(f(h_i, r_i, t_j)/\tau)} \quad (2)$$

where tail entity t_j traverses all entities in $\mathcal{E}$. The process is the same for the prediction of the head entity. To enhance the generalization ability, knowledge distillation methods transfer knowledge from the pre-trained teacher model to the student model (Hinton et al., 2015). The model training by optimizing an additional Kullback-Leibler (KL) divergence loss between the soft target probabilities from teacher and student in every batch:

$$\mathcal{L}_{KD} = \frac{1}{n}\sum_{i=1}^{n}\tau^2\cdot D_{KL}(\tilde{\mathbf{p}}_i^\tau||\mathbf{p}_i^\tau) \quad (3)$$

where $\tilde{\mathbf{p}}_i^\tau$ and $\mathbf{p}_i^\tau$ are the soft target probabilities, parameterized by temperature τ, from teacher and student models respectively. Using a higher value for temperature τ produces a softer probability distribution, resulting in a similar regularization effect as label smoothing (Yuan et al., 2020).

3.2. The Self-Knowledge Distillation module

For a clear notation, we denote the original batch of data sampled in the j^{th} iteration as $\mathcal{B}_j = \{(x_i, \mathbf{y}_i)\}_{i=1}^n$, and the network parameters as θ_j. Our model employs a data sampler to obtain $\mathcal{B}_j$ and $\mathcal{B}_{j+1}$ simultaneously in the j^{th} iteration. Both the predictions from $\mathcal{B}_j$ and $\mathcal{B}_{j+1}$ in the j^{th} iteration need to calculate the hard label loss. The hard label loss for our model is the original loss of the KGE method, usually a binary cross entropy loss. The $\mathbf{p}_i^\tau$ in (1) is denoted as $\mathbf{p}_i^{\tau,j}$ in the j^{th} iteration, which is formulated as:

$$\mathcal{L}_{Hard} = \frac{1}{n}\sum_{i=1}^{n}\tau^2\cdot\mathcal{L}_{BCE}(\mathbf{y}_i, \mathbf{p}_i^{\tau,j}) \quad (4)$$

where temperature τ is set to 1 in (4). To utilize the latest iteration information, our model replaces the $\tilde{\mathbf{p}}_i^\tau$ in (3) by the soft target probability $\mathbf{p}_i^{\tau,j-1}$ generated from $\mathcal{B}_j$ by the network with the same structure in the $(j-1)^{th}$ iteration, while the network

parameters are θ_{j-1}. The soft label loss for our model is the original loss of the knowledge distillation method, usually a KL divergence loss. The $\mathbf{p}_i^\tau$ in (3) is also denoted as $\mathbf{p}_i^{\tau,j}$ in the j^{th} iteration, which is formulated as:

$$\mathcal{L}_{Soft} = \frac{1}{n}\sum_{i=1}^{n} \tau^2 \cdot D_{KL}(\mathbf{p}_i^{\tau,j-1} || \mathbf{p}_i^{\tau,j}) \quad (5)$$

where temperature τ is set to a fixed value other than 1 in (5). Instead of storing the whole θ_{j-1} in the j^{th} iteration to calculate the soft target probabilities, which is run-in memory consuming. Our model obtains the soft target probability $\mathbf{p}_i^{\tau,j-1}$ from $\mathcal{B}_j$ in the $(j-1)^{th}$ iteration. In other words, predictions from $\mathcal{B}_j$ in the $(j-1)^{th}$ iteration are smoothed by temperature τ and then stored for regularization in the j^{th} iteration.

3.3. The Dynamic Temperature Distillation module

During model training, the distillation temperature is fixed. Distillation temperature may be high or low for different samples. At lower temperatures, distillation pays much less attention to matching predictions that are much more negative than the average. However, the very negative predictions may contain useful information about the knowledge (Hinton et al., 2015). At higher temperatures, the predictions may lose some discrimination information. Therefore, the model may be confused in some samples that get significant and similar predictions on several classes. As a solution, we use DTD (Wen et al., 2021), to customize the distillation temperature for each sample. DTD is used to make temperature τ vary on different training samples. Especially for samples that confuse easily, temperature τ should be smaller to enlarge inter-class similarity. And for easily learned samples, a bigger temperature τ will help to utilize the classification information about the classes. The modified soft label loss is as follows:

$$\mathcal{L}_{DTD} = \frac{1}{n}\sum_{i=1}^{n} \tau_d^2 \cdot D_{KL}(\mathbf{p}_i^{\tau_d,j-1} || \mathbf{p}_i^{\tau_d,j}) \quad (6)$$

where τ_d is sample-wise temperature in (6), and the calculation formula is as follows:

$$\tau_d = \tau_0 + (\frac{\sum_{i=1}^{n}\omega_i}{n} - \omega_d)\beta \quad (7)$$

where τ_0 and β denote the base temperature and bias in (7). And ω_d is sample-wise normalized weight, describing the extent of confusion. ω_i is used to calculate the average of ω_d over a batch. ω_d is designed to increase when the sample is confusing. Then τ_d will be smaller, and the soft target gets more discrimination information. On the opposite, ω_d is designed to decrease when the sample is easy to learn. Then τ_d will be larger, and the soft target gains more classification information. Here is how to calculate ω_d. The method computes ω_d following the max output of logits. The specific calculation method is as follows:

$$\omega_d = \frac{1}{\mathbf{s}_{max}} \quad (8)$$

where $\mathbf{s}_{max}$ represent the max output of logits produced by $\mathcal{B}_j$ in the j^{th} iteration. $\mathbf{s}_{max}$ can be deemed to represent the confidence of the model towards the sample. Samples with less confidence get larger weights and samples with more confidence get smaller weights.

3.4. The Knowledge Adjustment module

Although the model training process has been optimized, there is still a problem in the initial stage of model training. Early in model training, the accuracy of the soft target probabilities cannot be guaranteed. Incorrect predictions can mislead the training process, making training deviate from supervision. We then use KA (Wen et al., 2021), trying to fix the incorrect predictions with reference to the ground truth label. The exact operation is conducting an adjusting function $f_a(\cdot)$ on soft target probability $\mathbf{p}_i^{\tau,j-1}$. The modified soft label loss is as follows:

$$\mathcal{L}_{KA} = \frac{1}{n}\sum_{i=1}^{n} \tau_d^2 \cdot D_{KL}(f_a(\mathbf{p}_i^{\tau_d,j-1}) || \mathbf{p}_i^{\tau_d,j}) \quad (9)$$

where τ_d is sample-wise temperature in (9). Given an incorrect soft target, KA simply exchanges the value of ground truth (the theoretical maximum) and the value of predicted class (the predicted maximum), to ensure the maximum confidence is reached at the ground truth label. For example, we can see a soft target probability in Fig. 2. The ground truth label is "apple" but the prediction of a teacher is "pear". KA operation adjusts the theoretical maximum (apple) and the predicted maximum (pear). KA keeps all the soft target probabilities produced by $\mathcal{B}_j$ in the $(j-1)^{th}$ iteration instead of discarding the wrong ones. And the fixed tensor retains the overall numerical distribution of the soft target probabilities.

3.5. The SKDE module

Finally, we get SKDE after combining self-knowledge distillation, KA and DTD. The overall loss function is formulated as:

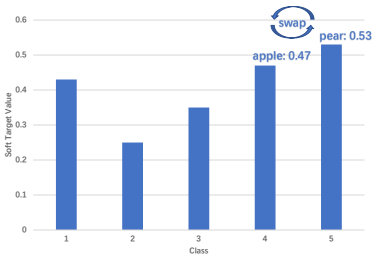

Figure 2: KA on a misjudged sample's soft target. In this case, where the correct answer is "apple," but the model's predicted probability for "pear" is the highest, it is necessary to correct this error to prevent misleading training of the model.

$$\begin{aligned}\mathcal{L} =& \frac{1}{n}\sum_{i=1}^{n}\tau^2 \cdot \mathcal{L}_{BCE}(\mathbf{y}_i, \mathbf{p}_i^{\tau,j}) \\ &+ \frac{1}{n}\sum_{i=1}^{n}\tau_d^2 \cdot D_{KL}(f_a(\mathbf{p}_i^{\tau_d,j-1})||\mathbf{p}_i^{\tau_d,j})\end{aligned} \tag{10}$$

4. Experiments

We evaluate the effectiveness and generalization ability of our SKDE by link prediction tasks. The purpose of the link prediction is to infer the possible missing entities in triples, predict h given (r, t) or predict t given (h, r).

Our experiments seek to answer the following research questions (RQs):

- RQ1: Can SKDE optimize model performance?
- RQ2: Can SKDE be lightweight without significantly degrading performance?
- RQ3: What role do KA and DTD play in SKDE?

4.1. Datasets and Settings

We use two public benchmark datasets for link prediction experiments, including WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova et al., 2015). Previous work (Dettmers et al., 2018; Toutanova et al., 2015) pointed out that doing prediction tasks on datasets WN18 and FB15K suffers from the problem of inverse relations. To solve this problem, corresponding subset datasets WN18RR and FB15k-237 are proposed. Table 1 provides statistics of all datasets used.

Dataset	WN18RR	FB15k-237
#entity	40,943	14,541
#relation	11	237
#train	86,835	272,115
#valid	3,034	17,535
#test	3,134	20,466
#total	93,003	310,116

Table 1: Statistics of datasets.

4.2. Baselines and Settings

To demonstrate the effectiveness and generalization ability of our SKDE on the link prediction task, we select different types of KGEs, such as:

- DistMult (Yang et al., 2015): a popular tensor factorization-based KGE model which uses a bi-linear scoring function to calculate the scores of knowledge triples.
- ComplEx (Trouillon et al., 2016): an advanced extension of DistMult which encodes entities and relations into complex vector space instead of real-valued vector space.
- ConvE (Dettmers et al., 2018): a popular convolutional network-based KGE model.
- AcrE (Ren et al., 2020): a simple but effective atrous convolution-based KGE model. There are two learning structures to integrate different kinds of convolutions: one is a serial structure, and the other is a parallel structure.

4.3. Training Protocol

In the training, we set the temperature $\tau_0 = 3$, coefficient $\beta = 1$. We set the learning rate $\mathrm{lr} = 0.003$, and decay the learning rate every epoch by $\text{lr-decay} = 0.995$. We set the dropout for the input embedding, convolutional feature and hidden layer by $\text{input-drop} = 0.2$, $\text{feat-drop} = 0.2$ and $\text{hidden-drop} = 0.3$, respectively. The initial dimensions of entities and relations embedding are set to $\text{embedding-dim} = 200$.

4.4. Evaluation Protocol

In the link prediction task, the goal is to predict the missing head or tail entities. For each testing triple, we remove the head entity or tail entity and replace it with each of the entities in $\mathcal{E}$ in turn. The model calculates a score for each triple and then sorts by descending order. As a result, we can get an accurate ranking of the correct triple in the candidates. The evaluation metrics include: the mean reciprocal rank (MRR) and the proportion of correct entities ranked in the top N (Hits@N, N=1, 3, 10).

Model	WN18RR				FB15k-237			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
DistMult	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
SKDE	**0.45**	**0.41**	**0.46**	**0.52**	**0.339**	**0.248**	**0.373**	**0.523**
SKDE (LW)	0.44	0.40	0.45	0.51	0.335	0.245	0.367	0.516
ComplEx	0.44	0.41	0.46	0.51	0.247	0.158	0.275	0.428
SKDE	**0.47**	**0.44**	**0.49**	**0.54**	**0.349**	**0.257**	**0.383**	**0.534**
SKDE (LW)	0.46	0.43	0.48	0.53	0.339	0.249	0.371	0.522
ConvE	0.43	0.40	0.44	0.50	0.311	0.223	0.339	0.493
SKDE	**0.44**	**0.41**	**0.46**	**0.53**	**0.326**	**0.236**	**0.356**	**0.508**
SKDE (LW)	**0.44**	0.40	0.45	0.52	0.324	0.234	0.355	0.506
AcrE(S)	0.44	0.40	0.45	0.51	0.324	0.244	0.363	0.481
SKDE	**0.47**	**0.43**	**0.48**	**0.55**	**0.341**	**0.249**	**0.375**	**0.525**
SKDE (LW)	0.43	0.38	0.46	0.52	0.334	0.245	0.367	0.511
AcrE(P)	0.45	0.42	0.46	0.52	0.328	0.247	0.367	0.485
SKDE	**0.48**	**0.44**	**0.49**	**0.55**	**0.353**	**0.260**	**0.390**	**0.540**
SKDE (LW)	0.41	0.35	0.44	0.51	0.339	0.250	0.372	0.517

Table 2: Link prediction results of KGEs and SKDEs on WN18RR and FB15k-237. SKDE (LW) is a lightweight SKDE. The embedding dimension of SKDE (LW) is decreased from 200 to 100.

4.5. Results and Analysis (RQ1)

To verify whether SKDE can optimize model performance, we first train these KGEs without making any changes to the network structure. Then, we add self-knowledge distillation, KA and DTD to different types of KGEs to get the corresponding SKDEs. We compare the performance of these models on the link prediction task.

We can find out from Table 2 that under the same embedding dimension, the model performance of each SKDE is significantly improved compared to the original KGE on all evaluation metrics. These results can prove the effectiveness and generalization ability of our SKDE.

4.6. The performance of SKDE after lightweight (RQ2)

The model size, i.e. the number of parameters is closely related to the embedding dimension of the entities and relations in the triple. To achieve a lightweight SKDE, we need to reduce the embedding dimension. On the premise of keeping the structure and hyperparameters of SKDE unchanged, we reduce the embedding dimension in half. The number of model parameters decreases as the embedding dimension decreases.

We can observe two conclusions from Table 3. Firstly, compared with the original KGE, the number of SKDE parameters does not change. This conclusion indicates that adding self-knowledge distillation, KA and DTD to the KGE does not change the number of model parameters. Secondly, the number of SKDE parameters decreases significantly as the embedding dimension is halved. This conclusion shows that reducing the embedding dimension can effectively reduce the number of model parameters. We can achieve the goal of the model being lightweight by reducing the embedding dimension. And then we explore the relationship between the embedding dimension and model performance.

Model	WN18RR	FB15k-237
DistMult	8,193,800	3,003,800
SKDE	8,193,800	3,003,800
SKDE (LW)	4,096,900	1,501,900
ComplEx	16,387,600	6,007,600
SKDE	16,387,600	6,007,600
SKDE (LW)	8,193,800	3,003,800
ConvE	10,181,299	4,964,897
SKDE	10,181,299	4,964,897
SKDE (LW)	4,599,299	1,977,897
AcrE(S)	10,813,329	5,596,927
SKDE	10,813,329	5,596,927
SKDE (LW)	4,796,529	2,175,127
AcrE(P)	11,435,873	6,219,471
SKDE	11,435,873	6,219,471
SKDE (LW)	4,938,873	2,317,471

Table 3: The number of model parameters. The number of parameters for each KGE method and the matching SKDE model are equal in all scenarios.

We can observe from Table 2 that after the embedding dimension of SKDE decreases, the perfor-

Model	WN18RR				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
SKDE (DistMult)	**0.446**	**0.408**	**0.463**	**0.521**	0.3396	0.2483	**0.3735**	**0.5234**
-DTD	0.442	0.407	0.458	0.512	0.3378	0.2475	0.3695	0.5199
-KA	0.443	0.407	0.457	0.516	0.3372	0.2472	0.3694	0.5198
-KA and DTD	0.442	**0.408**	0.456	0.507	**0.3402**	**0.2507**	0.3727	0.5193
SKDE (ComplEx)	**0.472**	**0.439**	**0.486**	**0.536**	**0.3493**	**0.2572**	**0.3839**	**0.5341**
-DTD	0.468	0.435	0.482	0.528	0.3443	0.2528	0.3781	0.5285
-KA	0.470	0.438	**0.486**	0.531	0.3474	0.2568	0.3811	0.5298
-KA and DTD	0.469	0.437	0.484	0.530	0.3452	0.2538	0.3789	0.5281
SKDE (ConvE)	**0.442**	**0.406**	**0.455**	**0.527**	**0.3258**	**0.2360**	**0.3563**	**0.5081**
-DTD	0.438	0.400	0.448	0.521	0.3211	0.2312	0.3511	0.5040
-KA	0.437	0.399	0.449	0.523	0.3234	0.2348	0.3519	0.5033
-KA and DTD	0.436	0.398	0.447	0.520	0.3217	0.2315	0.3537	0.5032
SKDE (AcrE(S))	**0.467**	**0.427**	**0.481**	**0.552**	**0.3412**	**0.2493**	**0.3754**	**0.5246**
-DTD	0.463	0.423	0.477	0.545	0.3356	0.2445	0.3694	0.5191
-KA	0.457	0.419	0.470	0.539	0.3368	0.2465	0.3692	0.5207
-KA and DTD	0.455	0.417	0.466	0.537	0.3364	0.2484	0.3718	0.5189
SKDE (AcrE(P))	**0.477**	**0.440**	**0.488**	**0.549**	**0.3529**	**0.2595**	**0.3900**	**0.5397**
-DTD	0.470	0.436	0.480	0.540	0.3492	0.2581	0.3826	0.5343
-KA	0.471	0.433	0.484	0.542	0.3486	0.2558	0.3830	0.5371
-KA and DTD	0.469	0.432	0.483	0.540	0.3490	0.2572	0.3880	0.5326

Table 4: Link prediction results of SKDEs, SKDEs without DTD, SKDEs without KA, SKDEs without KA and DTD on WN18RR and FB15k-237.

mance of SKDE (LW) shows a small decrease. In most cases, the lower the embedding dimension, the less information the model can learn, and the model performance will also decrease. However, our SKDE can maintain a good performance after the embedding dimension is reduced. The number of SKDE parameters also decreases as the embedding dimension decreases. Thus, we achieved our goal of obtaining a lightweight model while maintaining good performance. We also found that the performance of SKDEs after embedding dimension reduction can exceed the performance of KGEs without embedding dimension reduction. In this situation, we not only obtain a lightweight model but also achieve a performance improvement, which once again confirms the effectiveness and generalization ability of our SKDE.

We continue to explore the performance of SKDE as the model parameter decreases. We reduce the embedding dimension of lightweight SKDE from 100 to 50. Compared with the original KGEs with 200 embedding dimensions, the Hits@10 of lightweight SKDEs on WN18RR changed by +2.04%, +1.96%, +2.00%, +1.96%, -5.76%. The Hits@10 of lightweight SKDEs on FB15k-237 changed by +21.72%, +20.32%, -1.01%, -4.36%, -0.41%. From the above results, we can find that SKDE can still maintain the model performance after the embedding dimension is reduced to 50. Sometimes even better than the original KGE.

Then we discover the effects of model parameters on model training time. When the embedding dimension of SKDE is reduced from 200 to 100, the average training time is reduced by 4.13%. When the embedding dimension of SKDE is reduced from 200 to 50, the average training time is reduced by 12.00%. On the one hand, the lightweight model can reduce model training time. On the other hand, the lightweight model can reduce the consumption of computational sources and run-in memory by reducing the model parameters.

4.7. Ablation Study (RQ3)

The previous experimental results can fully prove the effectiveness and generalization ability of SKDE. Based on this, we further explore the role of KA and DTD in SKDE. We will remove KA and DTD from SKDE to observe the changes in model performance. We can discover from Table 4 that removing either KA or DTD degrades model performance, and removing KA and DTD together suffers the most.

From the experimental results in Table 4, we can draw the following conclusions: (i) Removing both KA and DTD is equivalent to having only self-knowledge distillation operating in the model. In this case, the model performance is still better

than the original KGE model. This shows that self-knowledge distillation alone can also be used to optimize model performance. (ii) Either KA or DTD alone can increase model performance. This indicates that both KA and DTD can optimize model performance in our model. (iii) There is a significant performance penalty to removing DTD from models. This suggests that DTD plays a more important role in our model than KA. (iv) Model performance is best when both KA and DTD are used. This shows that applying KA and DTD together to the model maximizes model performance.

5. Conclusion and Future Work

In this paper, we propose a method to apply self-knowledge distillation, KA and DTD to KGE, called SKDE. SKDE utilizes information from the latest iteration to guide training in the current iteration. Our method can fix the predictions of misjudged training samples. Our method also can design dynamic sample-wise temperatures to compute soft targets. Extensive experiments demonstrate the effectiveness and generalization ability of our SKDE. We achieve a lightweight model while maintaining a good model performance. The training time of the model is also decreased. In future work, we intend to pay more attention to the development of self-knowledge distillation. We will continue to optimize the self-knowledge distillation structure and explore the relationship between self-knowledge distillation and KGE.

6. References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? *Advances in neural information processing systems*, 27.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Dual quaternion knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6894–6902.

Tu Dinh Nguyen Dai Quoc Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of NAACL-HLT*, pages 327–333.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Liang Ding, Longyue Wang, Xuebo Liu, Derek F Wong, Dacheng Tao, and Zhaopeng Tu. 2021. Rejuvenating low-frequency words: Making the most of parallel data in non-autoregressive translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3431–3441.

Sangchul Hahn and Heeyoul Choi. 2019. Self-knowledge distillation in natural language processing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 423–430.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 978–987.

Kyungyul Kim, ByeongMoon Ji, Doyoung Yoon, and Sangheum Hwang. 2021. Self-knowledge distillation with progressive refinement of targets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6567–6576.

Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. 2020. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, pages 5714–5724. PMLR.

Yunshui Li, Junhao Liu, Chengming Li, and Min Yang. 2023. Self-distillation with meta learning for knowledge graph completion. *arXiv preprint arXiv:2305.12209*.

Yunteng Luan, Hanyu Zhao, Zhi Yang, and Yafei Dai. 2019. Msd: Multi-self-distillation learning via multi-classifiers within deep neural networks. *arXiv preprint arXiv:1911.09418*.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723.

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

Feiliang Ren, Juchen Li, Huihui Zhang, Shilei Liu, Bochao Li, Ruicheng Ming, and Yujia Bai. 2020. Knowledge graph embedding with atrous convolution and residual learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1532–1543.

Saed Rezayi, Handong Zhao, Sungchul Kim, Ryan Rossi, Nedim Lipka, and Sheng Li. 2021. Edge: Enriching knowledge graph embeddings with external text. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2767–2776.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bharat Bhusan Sau and Vineeth N Balasubramanian. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

Yiqing Shen, Liwu Xu, Yuzhe Yang, Yaqian Li, and Yandong Guo. 2022. Self-distillation from the last mini-batch for consistency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11943–11952.

George Stoica, Otilia Stretcu, Emmanouil Antonios Platanios, Tom Mitchell, and Barnabás Póczos. 2020. Contextual parameter generation for knowledge graph link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3000–3008.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2018. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1499–1509.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3009–3016.

Kai Wang, Yu Liu, Qian Ma, and Quan Z Sheng. 2021. Mulde: Multi-teacher knowledge distillation for low-dimensional knowledge graph embeddings. In *Proceedings of the Web Conference 2021*, pages 1716–1726.

Tiancheng Wen, Shenqi Lai, and Xueming Qian. 2021. Preparing lessons: Improve knowledge distillation with better supervision. *Neurocomputing*, 454:25–33.

Ting-Bing Xu and Cheng-Lin Liu. 2019. Data-distortion guided self-distillation for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5565–5572.

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*.

Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2020. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3903–3911.

Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. 2020. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13876–13885.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019a. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019b. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.

Yu Zhao, Han Zhou, Ruobing Xie, Fuzhen Zhuang, Qing Li, and Ji Liu. 2021. Incorporating global information in local attention for knowledge representation learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1341–1351.

Yushan Zhu, Wen Zhang, Mingyang Chen, Hui Chen, Xu Cheng, Wei Zhang, and Huajun Chen. 2022. Dualde: Dually distilling knowledge graph embedding for faster and cheaper reasoning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1516–1524.