Self-Distillation with Meta Learning for Knowledge Graph Completion

Yunshui Li[1,2]   **Junhao Liu**[1]   **Chengming Li**[3*]   **Min Yang**[1*]

[1]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]School of Intelligent Systems Engineering, Sun Yat-sen University
`{ys.li, jh.liu, min.yang}@siat.ac.cn, lichengming@mail.sysu.edu.cn`

Abstract

In this paper, we propose a self-distillation framework with meta learning (MetaSD) for knowledge graph completion with dynamic pruning, which aims to learn compressed graph embeddings and tackle the long-tail samples. Specifically, we first propose a dynamic pruning technique to obtain a small pruned model from a large source model, where the pruning mask of the pruned model could be updated adaptively per epoch after the model weights are updated. The pruned model is supposed to be more sensitive to difficult-to-memorize samples (e.g., long-tail samples) than the source model. Then, we propose a one-step meta self-distillation method for distilling comprehensive knowledge from the source model to the pruned model, where the two models co-evolve in a dynamic manner during training. In particular, we exploit the performance of the pruned model, which is trained alongside the source model in one iteration, to improve the source model's knowledge transfer ability for the next iteration via meta learning. Extensive experiments show that MetaSD achieves competitive performance compared to strong baselines, while being 10x smaller than baselines.

1 Introduction

Knowledge graphs (KG) have become the key technology to represent structural relations between entities and play an important role in question answering (Shen et al., 2018), dialogue systems (Yan et al., 2017), and entity disambiguation (Mulang' et al., 2020). However, most KGs are growing at a rapid pace and are far from complete. Therefore, it is necessary to develop knowledge graph completion (KGC) approaches to add missing triples to the KGs, so as to improve the quality of KGs.

Recent advances in KGC primarily work on knowledge graph embedding (KGE) by convert-
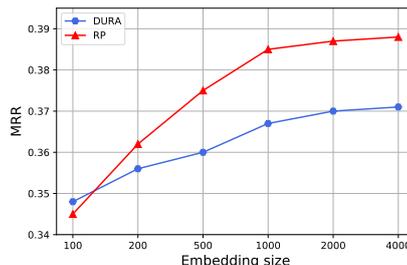
Figure 1: The MRR scores w.r.t. the graph embedding sizes of DURA and RP on FB15k-237.

ing the entities and relations in KGs into low-dimensional vectors. Early studies on KGE introduce a margin-based pairwise ranking function to measure the Euclidean distance or similarity between the relational projection of entities (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016). Among them, TransE (Bordes et al., 2013) is the most widely used KGE method, which views the relation as translation from a head entity to a tail entity. Recently, neural networks, such as neural tensor network (NTN) (Socher et al., 2013) and neural association model (NAM) (Liu et al., 2016) have been proposed to encode semantic matching and achieved remarkable predictive performance for KGC.

To increase the capacity of the KGE models, a larger embedding size with more parameters is a common technique in practice. As shown in Figure 1, the prediction performance of the KGC models such as DURA (Zhang et al., 2020) and RP (Chen et al., 2021) can be largely improved by increasing the graph embedding size. Although the large graph embedding often bring obvious performance improvements, it may also become the major obstacle for model deployment and real-time prediction, especially for memory-limited and resource-constrained devices.

In addition, the distribution of samples with long-tail is prevalent in KGs (Zhang et al., 2019), where a large portion of relations have much fewer triples than other relations. However, most previous stud-

---

ies mainly focus on the predictive performance on overall test data, without taking long-tail samples into consideration. These models suffer from robust and generalization performance in the practical scenario. Although several recent works (Xiong et al., 2018; Sheng et al., 2020) have been proposed for few-shot KGC, these models are not adapted to the model compression frameworks.

In this paper, we propose a self-distillation framework with meta learning (MetaSD) for knowledge graph completion with dynamic pruning. First, we propose a dynamic pruning technique to obtain a small pruned model from a large source model at the start of each training epoch. Concretely, the pruning mask of the pruned model could be updated adaptively per epoch after updating the model weights. The pruned model is supposed to be more sensitive to the difficult-to-memorize samples (e.g., long-tail samples) than the source model. Second, we propose a one-step meta self-distillation method to distill comprehensive knowledge from the source model to the pruned model, where the two models co-evolve in a dynamic manner during the whole knowledge distillation process. The key idea is to use the performance of the pruned model, which is trained alongside the source model in one iteration, to improve the source model for the next iteration by borrowing the idea of learning to learn from meta learning (Finn et al., 2017). In particular, we define the objectives of the source model as functions of the pruned model's performance on a quiz set. The usage of "gradient by gradient" strategy makes the source model adjust to the learning state of the pruned model, and improves both the source and pruned models.

The main contributions of our method can be three-fold. (1) We propose a self-distillation framework to compress KG embeddings for KGC. 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. (2) We exploit the feedback from the pruned model to guide the source model with meta learning, making the source model transfer better knowledge to the pruned network. (3) Experimental results on two benchmark datasets show that our model achieves competitive performance compared to strong baselines, while being 10x smaller than other KGC models.

2 Methodology

Problem Definition Suppose a KG can be viewed as a graph $\mathcal{G} = \{(h, r, t)\} \in E \times R \times E$, where E and R represent the entity (node) set and relation (edge) set respectively. (h, r, t) represents a triple, where h, t and r indicate head entity, tail entity and the relation between two entities, respectively. Given the KG $\mathcal{G}$, the goal of KGC is to infer missing links based on existing triples in the KG.

Model Overview The overview of our MetaSD method is illustrated in Figure 2. We adopt ComplEx (Trouillon et al., 2016) as our backbone model, which is treated as the source model $\mathcal{T}$. First, we use a magnitude-based weight pruning method (Han et al., 2015; Zhu and Gupta, 2017) to obtain a pruned model $\mathcal{S}$ from the source model $\mathcal{T}$. Second, we propose a one-step meta self-distillation method for distilling comprehensive knowledge from the source model $\mathcal{T}$ to the pruned model $\mathcal{S}$, where the two models co-evolve in a dynamic manner during training. Next, we introduce the proposed MetaSD method in detail.

2.1 Network Pruning

We use the magnitude-based weight pruning method (Han et al., 2015; Zhu and Gupta, 2017) to create a self-competitive compressed model $\mathcal{S}$ by pruning the source model $\mathcal{T}$. In particular, we fix the pruning rate γ during the whole training process. At each iteration, we first calculate the sum of parameter numbers of all layers be pruned as P, and sort all the weights by their absolute values. Then, we prune a certain fraction (i.e., γ) of weights that have lowest absolute weight values. In particular, to dynamically adjust the pruned network $\mathcal{S}$ during each iteration, we prune the chosen weight by setting the corresponding values in a binary mask to zero, instead of directly setting the weights to zero.

2.2 Self-Distillation via Meta Learning

We exploit the performance of the pruned model, which is trained alongside the source model in one iteration, to improve the source model's knowledge transfer ability for the next iteration via meta learning. In particular, we alternately update the pruned model $\mathcal{S}$ based on the output of the source model $\mathcal{T}$ and optimize the source model $\mathcal{T}$ based on the pruned model's performance via meta learning.
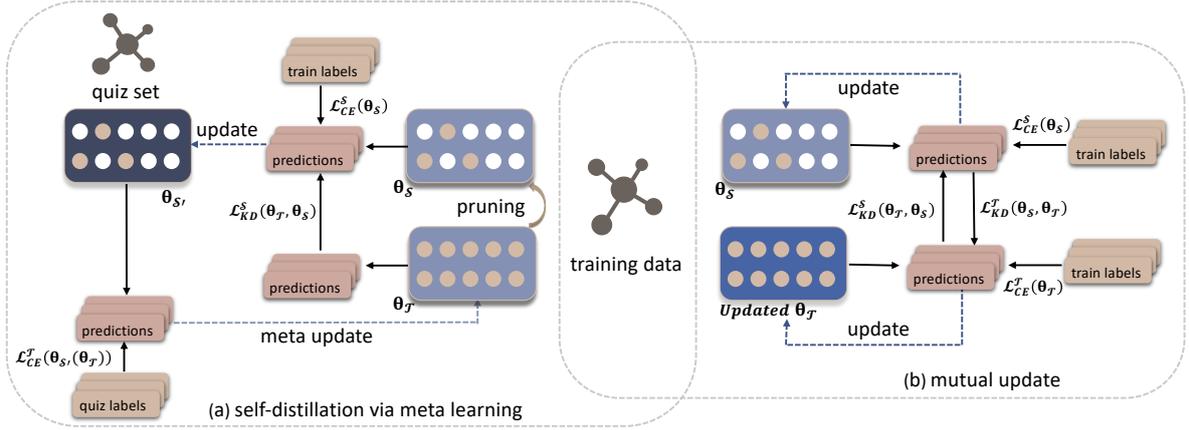
Figure 2: The overview of MetaSD framework. (a) We prune the teacher $\mathcal{T}$ to obtain the student $\mathcal{S}$ and perform knowledge distillation on training data to update the temporary copy $\mathcal{S}'$ from $\mathcal{S}$. Then, the source model $\mathcal{T}$ is optimized based on the feedback of $\mathcal{S}'$ on a held-out quiz set $\mathcal{Q}$; (b) We discard $\mathcal{S}'$ and optimize the meta-updated $\mathcal{T}$ and real $\mathcal{S}$ alternately by performing mutual learning on the training data.

Model $\mathcal{S}$ with Knowledge Distillation Formally, we use the function $\mathcal{S}(x_i; \theta_{\mathcal{S}})$ to denote the soft prediction of the compressed model, where $\theta_{\mathcal{S}}$ represents the parameters of the pruned model $\mathcal{S}$. We calculate the cross-entropy loss $\mathcal{L}_{\text{CE}}^{\mathcal{S}}(\theta_{\mathcal{S}})$ on the training data in current batch as:

$$\mathcal{L}_{\text{CE}}^{\mathcal{S}}(\theta_{\mathcal{S}}) = \frac{1}{N} \sum_{i=1}^{N} \text{CE}(y_i, \mathcal{S}(x_i; \theta_{\mathcal{S}})) \quad (1)$$

where N denotes the number of training samples. $\text{CE}(\cdot)$ represents the cross-entropy function.

To further improve the performance of $\mathcal{S}$, we also design a knowledge distillation loss $\mathcal{L}_{\text{KD}}^{\mathcal{S}}$ that encourages the output of $\mathcal{S}$ to mimic that of $\mathcal{T}$. In particular, we minimize the Kullback-Leibler Divergence (KL-divergence) between the output distributions of $\mathcal{S}$ and $\mathcal{T}$ by:

$$\mathcal{L}_{\text{KD}}^{\mathcal{S}}(\theta_{\mathcal{S}}, \theta_{\mathcal{T}}) = \frac{1}{N} \sum_{i=1}^{N} \text{KL}\big(\mathcal{S}(x_i; \theta_{\mathcal{S}})||\mathcal{T}(x_i; \theta_{\mathcal{T}})\big) \quad (2)$$

where $\theta_{\mathcal{T}}$ represents the parameters of $\mathcal{T}$.

The cross-entropy loss $\mathcal{L}_{\text{CE}}^{\mathcal{S}}$ and the knowledge distillation loss $\mathcal{L}_{\text{KD}}^{\mathcal{S}}$ are combined to form the overall loss $\mathcal{L}_{\mathcal{S}}$ for the compressed model $\mathcal{S}$ as:

$$\mathcal{L}_{\mathcal{S}}(\theta_{\mathcal{S}}, \theta_{\mathcal{T}}) = \alpha \mathcal{L}_{\text{CE}}^{\mathcal{S}}(\theta_{\mathcal{S}}) + (1 - \alpha)\mathcal{L}_{\text{KD}}^{\mathcal{S}}(\theta_{\mathcal{S}}, \theta_{\mathcal{T}}) \quad (3)$$

where α is a hyperparameter to balance the relative importance of the two loss functions.

Model $\mathcal{T}$ with Meta Learning We exploit feedback from the compressed model's learning state to improve the source model's knowledge transfer ability throughout the distillation process, instead of keeping the source model $\mathcal{T}$ fixed in the training process. We train both $\mathcal{T}$ and $\mathcal{S}$ in an iterative manner until convergence. This interaction between the two models can be seen as a form of meta learning with a bi-level optimization process, which com-

prises three steps: **Virtual-Train**, **Meta-Train**, and **Actual-Train** (Xu et al., 2021). That is, the compressed model $\mathcal{S}$ is the inner-learner and the source model $\mathcal{T}$ is the meta-learner.

For each training step, we first copy the parameters $\theta_{\mathcal{S}}$ of the compressed model $\mathcal{S}$ to a "virtual" compressed model $\mathcal{S}'$, and then update the parameters $\theta_{\mathcal{S}}'(\theta_{\mathcal{T}})$ of the "virtual" compressed model $\mathcal{S}'$ with SGD (Bottou, 2012) for the **Virtual-Train** as:

$$\theta_{\mathcal{S}}'(\theta_{\mathcal{T}}) = \theta_{\mathcal{S}} - \lambda \nabla_{\theta_{\mathcal{S}}} \mathcal{L}_{\mathcal{S}}(\theta_{\mathcal{S}}, \theta_{\mathcal{T}}) \quad (4)$$

Then, the source model $\mathcal{T}$ is optimized based on the feedback of $\mathcal{S}$ on a held-out quiz set $\mathcal{Q}$. We perform a derivative over a derivative (a Hessian matrix) to update $\theta_{\mathcal{T}}$, by using a retained computational graph of $\theta_{\mathcal{S}}'$ in order to compute derivatives with respect to $\theta_{\mathcal{T}}$. The source model $\mathcal{T}$ is optimized by minimizing the cross-entropy loss over the quiz set Q for the **Meta-Train** as:

$$\mathcal{L}_{\text{CE}}^{\mathcal{T}}\big(\theta_{\mathcal{S}}'(\theta_{\mathcal{T}})\big) = \frac{1}{M} \sum_{i=1}^{M} \text{CE}(y_i', \mathcal{S}(x_i'; \theta_{\mathcal{S}})) \quad (5)$$

where M is the training samples in the quiz set Q. x' and y' denote the input sample and corresponding label in quiz set $q \in \mathcal{Q}$, respectively.

Finally, we update the source model $\mathcal{T}$ with SGD (Bottou, 2012) as follows:

$$\theta_{\mathcal{T}} \leftarrow \theta_{\mathcal{T}} - \mu \nabla_{\theta_{\mathcal{T}}} \mathcal{L}_{\text{CE}}^{\mathcal{T}}(\theta_{\mathcal{S}}'(\theta_{\mathcal{T}})) \quad (6)$$

where μ is the learning rate for the Meta-Train.

Mutual Update of $\mathcal{T}$ and $\mathcal{S}$ for Self-Distillation In our self-distillation framework, the source model $\mathcal{T}$ and the compressed model $\mathcal{S}$ co-evolve in a dynamic manner during the whole KD process. Instead of updating $\mathcal{T}$ with cross-entropy loss, we learn both $\mathcal{T}$ and $\mathcal{S}$ models mutually.

Model	FB15k-237					WN18RR					Dim
	MRR	Hits@1	Hits@3	Hits@10	Size	MRR	Hits@1	Hits@3	Hits@10	Size	
TransE ⋆	0.313	0.221	0.347	0.497	-	0.228	0.053	0.368	0.520	-	-
RotatE ⋆	0.333	0.240	0.368	0.522	-	0.478	0.439	0.494	0.553	-	-
CP	0.333	0.247	0.360	0.508	50M	0.438	0.414	0.444	0.485	156M	2k
RESCAL	0.353	0.264	0.383	0.528	125M	0.455	0.419	0.460	0.493	26M	-
ComplEx	0.346	0.256	0.370	0.525	60M	0.460	0.428	0.475	0.522	156M	2k
DURA	0.371	0.276	0.408	0.560	60M	0.491	**0.449**	0.503	**0.571**	156M	2k
RP	0.388	0.298	0.425	0.568	60M	0.488	0.443	**0.505**	0.568	156M	2k
KD	0.371	0.282	0.408	0.550	6M	0.470	0.427	0.485	0.530	15M	0.2k
DML	0.373	0.280	0.410	0.563	6M	0.472	0.429	0.485	0.535	15M	0.2k
MetaSD	**0.391**	**0.300**	**0.428**	**0.571**	6M	0.491	<u>0.447</u>	<u>0.504</u>	<u>0.570</u>	15M	0.2k

Table 1: Experimental results on FB15k-237 and WN18RR test sets for KGC. The results with ⋆ are taken from LibKGE (Broscheit et al., 2020). CP, ComplEx and RESCAL are implemented by following (Zhang et al., 2020).

Formally, for the **Actual-Train**, we first update the compressed model's parameters $\theta_\mathcal{S}$ with the training data and the updated parameters $\theta_\mathcal{T}$ as:

$$\theta_\mathcal{S} = \theta_\mathcal{S} - \lambda \nabla_{\theta_\mathcal{S}} \mathcal{L}_\mathcal{S}(\theta_\mathcal{S}, \theta_\mathcal{T}) \qquad (7)$$

The source model $\mathcal{T}$ is also optimized by the combination of the cross-entropy loss $\mathcal{L}_{\mathrm{CE}}^\mathcal{T}$ and the knowledge distillation loss $\mathcal{L}_{\mathrm{KD}}^\mathcal{T}$ as:

$$\mathcal{L}_{\mathrm{CE}}^\mathcal{T}(\theta_\mathcal{T}) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{CE}(y_i, \mathcal{T}(x_i; \theta_\mathcal{T})) \qquad (8)$$

$$\mathcal{L}_{\mathrm{KD}}^\mathcal{T}(\theta_\mathcal{S}, \theta_\mathcal{T}) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{KL}\big(\mathcal{S}(x_i; \theta_\mathcal{S}) || \mathcal{T}(x_i; \theta_\mathcal{T})\big) \qquad (9)$$

$$\mathcal{L}_\mathcal{T}(\theta_\mathcal{S}, \theta_\mathcal{T}) = \beta \mathcal{L}_{\mathrm{CE}}^\mathcal{T}(\theta_\mathcal{S}) + (1 - \beta) \mathcal{L}_{\mathrm{KD}}^\mathcal{T}(\theta_\mathcal{S}, \theta_\mathcal{T}) \qquad (10)$$

where β is a hyperparameter to balance the relative importance of the two loss functions. We first update the source model's parameters $\theta_\mathcal{T}$ with the training data and the updated parameters $\theta_\mathcal{S}$ as:

$$\theta_\mathcal{T} = \theta_\mathcal{T} - \lambda \nabla_{\theta_\mathcal{T}} \mathcal{L}_\mathcal{T}(\theta_\mathcal{S}, \theta_\mathcal{T}) \qquad (11)$$

We train the source and compressed models in an iterative manner until convergence. Overall, the self-distillation with meta learning is defined in Algorithm 1.

Algorithm 1 Self-Distillation with Meta Learning

Require: train set $\mathcal{D}$, quiz set $\mathcal{Q}$, source model $\theta_\mathcal{T}$
Require: learning rate λ, learning rate μ, $i \leftarrow 0$
1: **repeat**
2:     $i \leftarrow i + 1$
3:     Sample a batch of training data x from $\mathcal{D}$
4:     Get pruned model $\theta_\mathcal{S}$ by pruning $\theta_\mathcal{T}$
5:     Copy pruned model's parameters $\theta_\mathcal{S}$ to a "virtual" pruned model $\theta_\mathcal{S}'$
6:     Update $\theta_\mathcal{S}'$ with x and $\theta_\mathcal{T}$: # *Virtual-Train*
      $\theta_\mathcal{S}'(\theta_\mathcal{T}) = \theta_\mathcal{S} - \lambda \nabla_{\theta_\mathcal{S}} \mathcal{L}_\mathcal{S}(x; \theta_\mathcal{S}, \theta_\mathcal{T})$
7:     Sample a batch of quiz data q from $\mathcal{Q}$
8:     Update $\theta_\mathcal{T}$ with q and $\theta_\mathcal{S}'$: # *Meta-Train*
      $\theta_\mathcal{T} \leftarrow \theta_\mathcal{T} - \mu \nabla_{\theta_\mathcal{T}} \mathcal{L}_{\mathrm{CE}}^\mathcal{T}(q; \theta_\mathcal{S}'(\theta_\mathcal{T}))$
9:     Mutual update $\theta_\mathcal{T}$ and $\theta_\mathcal{S}$: # *Actual-Train*
      $\theta_\mathcal{S}(\theta_\mathcal{T}) = \theta_\mathcal{S} - \lambda \nabla_{\theta_\mathcal{S}} \mathcal{L}_\mathcal{S}(x; \theta_\mathcal{S}; \theta_\mathcal{T})$
      $\theta_\mathcal{T}(\theta_\mathcal{S}) = \theta_\mathcal{T} - \lambda \nabla_{\theta_\mathcal{T}} \mathcal{L}_\mathcal{T}(x; \theta_\mathcal{S}; \theta_\mathcal{T})$
10: **until** $i ==$ *max iterations*
Output: source model $\theta_\mathcal{T}$ and pruned model $\theta_\mathcal{S}$

---

3 Experimental Setup

3.1 Datasets

We conduct experiments on two KGC benchmark datasets: WN18RR (Toutanova and Chen, 2015) and FB15k-237 (Dettmers et al., 2018). WN18RR consists of 40,943 entities and 11 relations, and there are 86k/3k/3k instances for training/validation/testing respectively. FB15k-237 contains 14,541 entities and 237 relations, and there are 272k/17k/20k instances for training/validation/testing.

3.2 Baseline Methods

We compare MetaSD with several strong KGC baselines, including CP (F.L, 1927), RESCAL (Nickel et al., 2011), TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019), DURA (Zhang et al., 2020), and RP (Chen et al., 2021). We also compare MetaSD with two widely used KD methods: knowledge distillation (KD) (Hinton et al., 2015) and deep mutual learning (DML) (Zhang et al., 2018), where the pre-trained RP (Chen et al., 2021) is used as their teacher model.

3.3 Implementation Details

The source model of MetaSD is initialized with ComplEx (Trouillon et al., 2016), following the previous work (Zhang et al., 2020). Similar to Chen et al. (2021), we add relation prediction as an auxiliary task. We set the pruning rate γ to 0.9 to strike a balance between the effectiveness and efficiency of the model. We set balance hyperparameters $\alpha = \beta = 0.5$. We choose Adagrad (Duchi et al., 2011) as the optimizer and the learning rate μ to $1e^{-4}$ and λ to $1e^{-1}$. The quiz set is randomly sampled from training data and then fixed. We adopt widely used *filtered* evaluation metrics of mean reciprocal rank (MRR), Hits@1, Hits@3,

Model	MRR	H@1	H@3	H@10
DURA	0.452	0.354	0.498	0.644
RP	0.462	0.372	0.504	0.645
MetaSD-$\mathcal{T}$	0.468	0.380	0.510	0.642
MetaSD	**0.471**	**0.381**	**0.512**	**0.646**

Table 2: Results on long-tail data from FB15k-237.

Model	MRR	H@1	H@3	H@10
MetaSD	**0.391**	**0.300**	**0.428**	**0.571**
w/o P	0.381	0.292	0.415	0.561
w/o M	0.378	0.287	0.412	0.555
w/o P&M	0.373	0.280	0.410	0.563

Table 3: Results of ablation study on FB15k-237. P and M denote the pruning and meta learning techniques.

and Hits@10 as described in (Bordes et al., 2013).

4 Experimental Results

4.1 Overall Performance

As shown in Table 1, we report the results of the compressed model (denoted as MetaSD). Note that the parameters of RASCAL are proportional to the square of the number of relations, resulting in large differences in size between the two datasets. We observe that MetaSD achieves competitive performance compared to other high-dimensional baseline models on the two datasets for KGC, while being 10x smaller than baseline methods. In addition, MetaSD also outperforms than two widely used KD methods that have the same size and dimension with MetaSD.

4.2 Long-tail Evaluation

We investigate the effectiveness of MetaSD for dealing with the long-tail samples. In particular, we collect the long-tail samples from the FB15k-237 test set by choosing the relations that have fewer than 1000 training instances. In total, there are 187 relations, which accounts for 79% of the total relation types but only 24% of train set. Table 2 reports the results of MetaSD and compared methods on the long-tail set. MetaSD significantly outperforms other models on the long-tail samples, which verifies the effectiveness of MetaSD in tackling the long-tail samples.

4.3 Ablation Study

In order to verify the effectiveness of the pruning and meta learning modules, we conduct ablation evaluation on the proposed MetaSD on the FB15k-237 dataset. As shown in Table 3, we can observe that the meta learning technique has great impact on the performance of the proposed MetaSD. This

Model	MRR	H@1	H@3	H@10	Size
CP	0.333	0.247	0.360	0.508	50M
RESCAL	0.353	0.264	0.383	0.528	125M
MetaSD-CP	0.367	0.270	0.396	0.557	5M
MetaSD-RESCAL	0.372	0.276	0.405	0.561	12.5M

Table 4: Results of MetaSD on FB15k-237 by using different backbone models.

is because that meta learning can make the source network transfer rich knowledge to the pruned network effectively in the self-distillation process. In addition, the improvement of the self-pruning strategy is also significant since self-pruning can help the model learn discriminative representations and deal with the long-tail samples. It is no surprise that combining both factors achieves the best performance on in terms of four evaluation metrics.

4.4 Generalization

To demonstrate the robustness of our framework, we also implement MetaSD on two additional backbone models (e.g., CP and RESCAL). These two backbone models are implemented and initialized by following the paper (Zhang et al., 2020). As shown in Table 4, our compressed model achieves substantially better performance than the larger baseline models based on two different backbone models.

5 Conclusion

In this paper, we proposed a self-distillation framework with meta learning for graph embedding compression. Concretely, we proposed a one-step meta self-distillation method for distilling comprehensive knowledge from the source model to the pruned model, where the two models co-evolved in a dynamic manner during training. Experimental results showed that our model achieved competitive performance compared to strong baseline methods, while being 10x smaller than baseline methods.

Acknowledgements

Limitations

To better understand the limitations of the proposed model, we carry out an analysis of the error predictions made by MetaSD. In particular, we primarily analyze the relations in the FB15k-237 test set, whose MRR scores are less than 0.2. Most of the incorrectly predicted relations are the "location" and "relationships" related relation types, such as place of birth/death, spouse, and sibling. We reveal several reasons of the bad cases, which can be divided into two primarily categories. First, MetaSD fails to predict some instances that require the multi-hop reasoning to get the correct answers, since our model does not consider the complex multi-hop paths during the knowledge graph representation learning. Second, MetaSD fails to predict some instances, where there are a large number of candidate entities to reason for a relation type (e.g., the location relation). One possible solution is to devise a two-step ranking method by filtering most of the irrelevant entities in a coarse-grained way and then distinguish the confusing entities with a fine-grained method.

References

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.

Léon Bottou. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. LibKGE - A knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 165–174.

Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. 2021. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. *arXiv preprint arXiv:2110.02834*.

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.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

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

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

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

Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. 2016. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*.

Isaiah Onando Mulang', Kuldeep Singh, Chaitali Prabhu, Abhishek Nadgeri, Johannes Hoffart, and Jens Lehmann. 2020. Evaluating the impact of knowledge graph context on entity disambiguation models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2157–2160.

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

Ying Shen, Yang Deng, Min Yang, Yaliang Li, Nan Du, Wei Fan, and Kai Lei. 2018. Knowledge-aware attentive neural network for ranking question answer pairs. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 901–904.

Jiawei Sheng, Shu Guo, Zhenyu Chen, Juwei Yue, Lihong Wang, Tingwen Liu, and Hongbo Xu. 2020. Adaptive attentional network for few-shot knowledge graph completion. *arXiv preprint arXiv:2010.09638*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *Advances in neural information processing systems*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

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.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-shot relational learning for knowledge graphs. *arXiv preprint arXiv:1808.09040*.

Youjiang Xu, Linchao Zhu, Lu Jiang, and Yi Yang. 2021. Faster meta update strategy for noise-robust deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 144–153.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Thirty-first AAAI conference on artificial intelligence*.

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

Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. *arXiv preprint arXiv:1903.01306*.

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.

Zhanqiu Zhang, Jianyu Cai, and Jie Wang. 2020. Duality-induced regularizer for tensor factorization based knowledge graph completion. *Advances in Neural Information Processing Systems*, 33:21604–21615.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.