Weighted Knowledge Graph Embedding

Zhao Zhang
Institute of Computing Technology, Chinese Academy of Sciences
zhangzhao2021@ict.ac.cn

Zhanpeng Guan
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences

Fuwei Zhang
Institute of Computing Technology, Chinese Academy of Sciences
University of Chinese Academy of Sciences

Fuzhen Zhuang*
Institute of Artificial Intelligence, Beihang University
Zhongguancun Laboratory
zhuangfuzhen@buaa.edu.cn

Zhulin An*
Institute of Computing Technology, Chinese Academy of Sciences
anzhulin@ict.ac.cn

Fei Wang
Yongjun Xu
Institute of Computing Technology, Chinese Academy of Sciences
{wangfei,xyj}@ict.ac.cn

ABSTRACT

Knowledge graph embedding (KGE) aims to project both entities and relations in a knowledge graph (KG) into low-dimensional vectors. Indeed, existing KGs suffer from the data imbalance issue, i.e., entities and relations conform to a long-tail distribution, only a small portion of entities and relations occur frequently, while the vast majority of entities and relations only have a few training samples. Existing KGE methods assign equal weights to each entity and relation during the training process. Under this setting, long-tail entities and relations are not fully trained during training, leading to unreliable representations. In this paper, we propose WeightE, which attends differentially to different entities and relations. Specifically, WeightE is able to endow lower weights to frequent entities and relations, and higher weights to infrequent ones. In such manner, WeightE is capable of increasing the weights of long-tail entities and relations, and learning better representations for them. In particular, WeightE tailors bilevel optimization for the KGE task, where the inner level aims to learn reliable entity and relation embeddings, and the outer level attempts to assign appropriate weights for each entity and relation. Moreover, it is worth noting that our technique of applying weights to different entities and relations is general and flexible, which can be applied to a number of existing KGE models. Finally, we extensively validate the superiority of WeightE against various state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies → Knowledge representation and reasoning**.

KEYWORDS

Knowledge Graph Embedding, Bilevel Optimization, Link Prediction

*Corresponding authors: Fuzhen Zhuang and Zhulin An

SIGIR '23, July 23–27, 2023, Taipei, Taiwan
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3591784

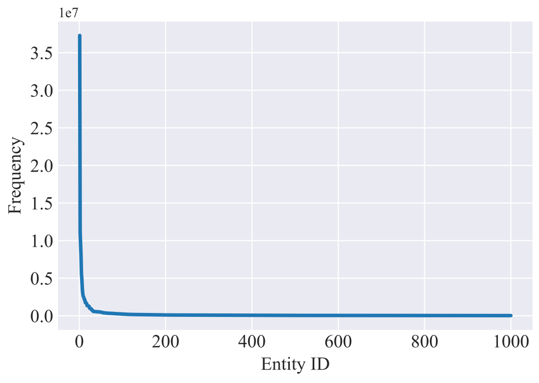
ACM Reference Format:
Zhao Zhang, Zhanpeng Guan, Fuwei Zhang, Fuzhen Zhuang, Zhulin An, Fei Wang and Yongjun Xu. 2023. Weighted Knowledge Graph Embedding. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3539618.3591784

1 INTRODUCTION

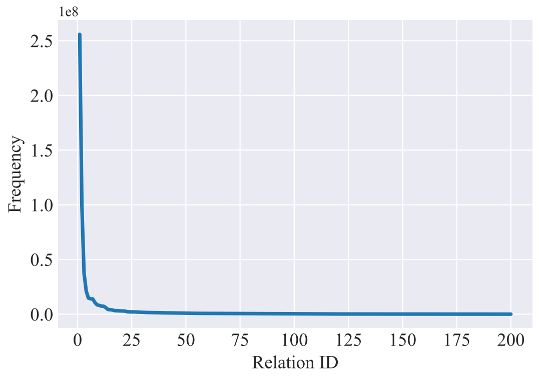
Knowledge Graphs (KGs), such as Freebase [2], Yago [36] and Wikidata [41] have been pivotal resources for a number of AI-related applications, including recommender systems [15, 58], information retrieval [35, 50], question answering [17, 27], time series forecasting [10], and drug discovery [48]. Indeed, KGs are multi-relational directed graphs composed of entities as nodes and relations as different types of edges. The information of real-world entities and relations is modeled in the form of knowledge triples, which are denoted as (s, r, o), where s and o are two entities, corresponding to the subject and object of the triple, and r denotes the relation between them, e.g., (*London, capital_of, UK*).

Although knowledge triples represent real-world facts in an easy-to-understand manner, their symbolic nature makes it hard for them to be used by machine learning, especially deep learning algorithms [56]. Under the circumstances, knowledge graph embedding (KGE) methods quickly attracted massive attention. KGE aims to project both entities and relations into a continuous low-dimensional space, so as to simplify the manipulation while preserving the inherent structure of the KG. Such embeddings encode rich information of entities and relations, and can be widely utilized in neural network based models for knowledge completion, fusion and inference [44, 54].

However, existing KGs suffer from the data imbalance problem. Figure 1 shows the frequencies of entities and relations in the real-world KG Wikidata [41], where the vertical axis represents the frequency, and the horizontal axis denotes entity and relation IDs sorted by the corresponding frequencies. We find entities and relations conform to a long-tail distribution, i.e., only a small number of entities and relations occur frequently, while the vast majority of entities and relations are only associated with limited triples. For better visualization, we only show top 1000 frequent entities and top 200 frequent relations. Note that there are more than 9×10^7 entities and more than 1000 relations in Wikidata. Thus the real



(a) Entity Frequency



(b) Relation Frequency

Figure 1: Entity and relation frequency in the real-world knowledge graph Wikidata.

situation is much more severe than the figure shows. Nevertheless, most existing methods ignore the data imbalance issue, resulting in unreliable representations for long-tail entities and relations[1]. For preliminary experiments, we run the widely used KGE model TransE [3] on the benchmark dataset FB15k-237 [38]. We find for test triples with the frequent entity *United Kingdom* which has 1953 occurrences, TransE achieves the mean reciprocal rank (MRR) score of 0.408 on the link prediction task. While for test triples with the infrequent entity *Pisa* which only occurs 8 times, TransE obtains a much lower MRR score of 0.194, which is in line with our conjecture. Thus it is critical to learn more reliable representations for infrequent entities and relations, which is beneficial for KGE models to achieve better performance in downstream tasks.

To this end, in this paper, we aim to learn better representations for long-tail entities and relations during the training process. In the current KGE setting, existing models do not differentiate the weights of different entities and relations, and thus tend to pay more attention to the frequent ones due to their high frequency. Current approaches that tackle the data imbalance issue focus on the few-shot link prediction (a.k.a. KG completion) task, in which the researchers apply the few-shot setting to predict missing values in KG. Under this setting, each few-shot learning task corresponds to an infrequent relation that only occurs a few times (e.g., 5-shot). However, existing methods have the following two flaws. (1) The few-shot setting is rather too strict, in which each low-frequency relation has only a certain number of occurrences (e.g., 5). And this may not hold in practice. As we know in real-world KGs, the number of occurrences of entities and relations varies widely and does not follow a specific number. (2) Most existing works only focus on the data imbalance problem of relations, while paying no attention to the long-tail distribution of entities, which is also critical in practice.

Different from existing methods, we propose WeightE, which is capable of well tackling the data imbalance issue via attending differentially to different entities and relations. Specifically, WeightE assigns different weights to different entities and relations during the training procedure. A straightforward idea is to set a weight parameter for each entity and relation, and update the weights and embeddings simultaneously during the training process of KGE models. However, we find this naive approach is not appropriate in our case due to the following two reasons. (1) Previous works [4, 28] have shown that simultaneously learning the representations and weights for training data may hinder both learning processes, leading to unsatisfactory performance. (2) A number of state-of-the-art KGE models utilize hinge loss or softplus loss as the objective function. Under this setting, the gradient direction of entities' and relations' weights is single and fixed, severely hindering the learning process. To this end, in this paper, we utilize the technique of bilevel optimization to reweight the entities and relations during the training procedure. In particular, WeightE specially tailors bilevel optimization for the KGE task, where the goal of the inner loop is to learn reliable entity and relation representations, and the outer loop aims to assign appropriate weights for each entity and relation. In this way, WeightE is capable of endowing lower weights to frequent entities and relations, and higher weights to low-frequency ones. Under this setting, KGE models are able to pay more attention to long-tail entities and relations, and learn better representations for them. It is worth noting that the technique of reweighting entities and relations is general and flexible, which can be applied to a number of existing KGE models, e.g., TransE [3], DistMult [46], ConvKB [24], and RotatE [37]. Finally, we conduct extensive experiments on benchmark datasets, and results show that the proposed method is able to appropriately reweight entities and relations, and consistently outperform state-of-the-art competitors.

In a nutshell, we highlight our key contributions as follows.

- In this paper, we propose a novel model WeightE, which specially tailors the technique of bilevel optimization to alleviate the data imbalance issue in KGE.
- The proposed reweighting technique is general and flexible, which can be applied to a number of existing KGE models.
- Experiments on real-world datasets show that the proposed method is able to assign appropriate weights for each entity and relation, and outperform state-of-the-art KGE models.

[1]In this paper, long-tail entities/relations indicate entities/relations with low frequencies.

2 RELATED WORK

2.1 Knowledge Graph Embedding

Recent years have witnessed the increasing interest in KGE, which aims to represent entities and relations in KGs as low-dimensional vectors. Existing works roughly fall into three categories.

Geometric methods. These methods utilize geometric operations to represent entities and relations. TransE [3] is one of the most widely used KGE models, which assumes $v_s + v_r \approx v_o$ when (s, r, o) holds. v_s, v_r and v_o are embedding vectors of s, r, and o, respectively. TransH [43], TransR [19], RotatE [37], Rotate3D [14], and HAKE [53] also fall into this category.

Tensor decomposition methods. These models assume the score of a (s, r, o) triple can be factorized into several tensors [39, 46]. DistMult [46] is a representative model in the category, which decomposes the score of a triple into two vectors and a diagonal matrix. ComplEx [39], HolE [25] and ANALOGY [20] further extend DistMult to achieve better performance.

Neural network methods. These models take advantage of deep neural networks to represent entities and relations in KGs. ConvE [11] and ConvKB [24] adopt convolutional neural network (CNN) to capture the interactions between entities and relations. R-GCN [31], RGHAT [57] and CompGCN [40] utilize graph neural network (GNN) to learn KG embeddings.

There is a line of research works that try to embed uncertain KGs. Note that these works are different from ours. These works focus on learning entity and relation representations in uncertain KGs, in which each triple has a ground truth confidence score, indicating the plausibility of the triple. The higher the score, the more weights the triple plays during the training procedure. While our work focuses on general KGs, which do not have ground truth weights for each entity and relation.

2.2 Data Imbalance Issue in KGE

Although the data imbalance issue in KG has existed for a long time, related work that aims to tackle this issue is very limited. Most existing works focus on the link prediction task in KG, in which each relation only has a limited number of occurrences (e.g., 5-shot). This task aims to predict the missing links for relations that only have a few associative triples. Existing few-shot link prediction models can be divided into two categories: metric learning based models [45, 49] and meta learner based models [5, 26]. However, the above methods have the following two flaws, which hinder them to be applied to KGs in real-world scenarios. (1) The few-shot setting is too strict, requiring each infrequent relation to have a certain number of occurrences, which may not hold in practice. (2) These models only focus on the data imbalance issue for relations, while ignoring the long-tail distribution of entities.

Apart from the above methods, related work is rather rare. TranSparse [16] introduces adaptive sparse matrices to facilitate better learning for infrequent relations. wRAN [51] utilizes an adversarial procedure to help to adapt features learned from high-frequency relations to low-frequency ones. These two works only focus on the data imbalance issue of relations. The most relevant work is LSU [55], which transfers knowledge from high-frequency entities/relations to infrequent ones via sharing latent semantic units. Different from existing methods, in this paper, we reweight entities and relations, and lift up the weights of the long-tail ones during the training process.

2.3 Bilevel Optimization

Bilevel optimization is defined as a mathematical paradigm, where an optimization problem contains another optimization problem as a constraint [33]. Bilevel optimization is able to optimize two closely related objectives in an alternate manner [8]. Recently, this technique has become a timely and important topic due to its great effectiveness in hyperparameter optimization, meta-learning, and reinforcement learning [47]. Indeed, bilevel optimization has been applied to a number of real-world applications. λOPT [7] uses bilevel optimization to learn regularized terms in recommender systems. Franceschi et al. [13] introduce a framework based on bilevel optimization that unifies gradient-based hyperparameter optimization and meta-learning for image classification. Chen et al. [6] apply bilevel optimization to a number of NLP tasks. The combination of bilevel optimization with KGE is limited, AutoSF [52] uses bilevel optimization to search the score function of knowledge triples. In this way, the score function of a KGE model is decided in an automatic manner instead of a hand-designed manner. In this paper, we make use of bilevel optimization to learn the weights of different entities and relations in KGE.

3 PRELIMINARIES

In this section, we provide some basic definitions used in this paper.

Definition 1. ***Knowledge Graph.*** *A knowledge graph is viewed as a graph $\mathcal{G} = \{(s, r, o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the entity (node) set and relation (edge) set, respectively.*

Definition 2. ***Frequent/Infrequent Entities/Relations.*** *In a KG, the entities with top 20% frequencies are named as frequent/high-frequency entities, while the remaining 80% entities are infrequent/low-frequency entities. Frequent/Infrequent relations are defined in a similar way.*

Definition 3. ***Frequency-aware Triples.*** *For a (s, r, o) triple, if s and o are frequent entities, and r is a frequent relation; or s and o are infrequent entities, and r is an infrequent relation, this triple is named as frequency-aware triples.*

4 METHODOLOGY

In this section, we discuss the role that reweighting plays on KGE. Specifically, we first detail the reweighting technique of WeightE. Then we present how to apply the technique to existing KGE models. Finally, we provide optimizer choices to train WeightE.

4.1 Reweighting Technique on KGE

4.1.1 Overview. Existing KGs suffer from the data imbalance issue. Specifically, entities and relations conform to a long-tail distribution. And the long-tail entities and relations cannot be fully trained during the training process, leading KGE models to learn unreliable representations for them. The reweighting technique has been shown to be an effective way to tackle the data imbalance issue [32, 34]. However, directly utilizing previous reweighting methods to KGE is not appropriate. Specifically, we aim to reweight entities and relations in KG, but the smallest training unit during

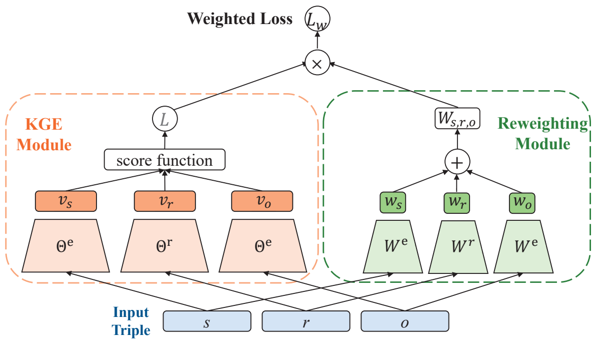


Figure 2: An illustration of the weighted loss.

KGE training is a triple rather than an entity or a relation. To this end, we propose the following reweighting method for KGE,

$$L_w(\Theta \mid W) = \sum_{(s,r,o)\in S_T} W_{(s,r,o)} \cdot L(\Theta_{(s,r,o)}), \tag{1}$$

$$W_{(s,r,o)} = \frac{1}{3}\left(w_s + w_r + w_o\right), \tag{2}$$

where S_T is the training set. w_s, w_r, and w_o are the corresponding weights for subject, relation, and object, respectively. $W_{(s,r,o)}$ is the weight for the triple (s, r, o). Θ represents the parameters of the KGE model, and $\Theta_{(s,r,o)}$ denotes the parameters that are related to the triple (s, r, o). L is the original loss function of the KGE model, and L_w is the corresponding reweighted version. To facilitate a better understanding, we provide an illustration of the weighted loss, which is shown in Figure 2. It is worth noting that the method is general and flexible, and can be applied to a number of existing KGE models via changing L. In the following, for simplicity, we omit the subscript of W and Θ, thus Eq. (1) changes to $L_w(\Theta \mid W) = \sum_{(s,r,o)\in S_T} W \cdot L(\Theta)$.

In order to appropriately train the weights of entities and relations W and KGE parameters Θ, we formulate the training process as a bilevel optimization problem, which is shown as

$$\min_W \sum_{(s,r,o)\in S_O} L\left(v_s, v_r, v_o \mid \arg\min_\Theta \sum_{(s,r,o)\in S_I} L_w(v_s, v_r, v_o \mid \Theta, W)\right). \tag{3}$$

S_I and S_O are the parts of the training set used to train the inner and outer loops, respectively, i.e., $S_I \cup S_O = S_T$. Specifically, before each training epoch, we randomly select 80% of the training data as S_I, and the remaining 20% as S_O. During the training process, the inner level aims to learn reliable entity and relation embeddings with the inner set S_I and the weighted loss L_w, and the outer level targets at learning appropriate weights for each entity and relation with the outer set S_O and the original KGE loss L.

Note that we only update the weights for frequency-aware triples (cf. Definition 3 in Section 3). For other triples, we set the weights of entities and relations as 1. We adopt this setting based on the following reason. According to Eq. (1) and Eq. (2), for a triple (s, r, o), the gradients of w_s, w_r and w_o are in the same direction, i.e., the values of weights are increased together or decreased together during training. For triples that are comprised of frequent entities and relations, we expect w_s, w_r and w_o to decrease. While for triples that consist of infrequent entities and relations, we expect w_s, w_r and w_o to increase. However, for triples other than the frequency-aware ones, we expect some values of w_s, w_r and w_o to increase, and some values to decrease. Simultaneously increasing or decreasing w_s, w_r and w_o would negatively affect some of the weight values, which may lead to unsatisfactory results. Thus we set the weights of entities and relations in such triples as 1.

4.1.2 *Reweighting Strategy.*

In this subsection, we provide two reweighting approaches for KGE.

(1) *Fixed Reweighting Approach.* To set different weights for each entity and relation, a naive idea is to manually set a fixed weight for each entity/relation. One of the methods is to sort the entities/relations in descending order according to frequency, and select the frequency of the entity and relation ranked 20% as a pivot. Then the weights of entities and relations can be defined as

$$w_e = \frac{f_e^{pivot}}{f_e}, w_r = \frac{f_r^{pivot}}{f_r}, \tag{4}$$

where f_e and f_r denote the frequency of entity e and relation r. f_e^{pivot} and f_r^{pivot} are the pivot frequency for entity and relation. Under this setting, the weights of high-frequency entities/relations are lowered, and the weights of low-frequency ones can be lifted.

However, there are a large number of methods to set the fixed weights, and experimenting with them one by one is time-consuming. In addition, the fixed setting greatly reduces the model flexibility. Therefore, we propose an adaptive update method in the next part.

(2) *Adaptive Reweighting Approach.* Instead of the fixed reweighting approach, the adaptive approach can automatically adjust the weights during iteration. A straightforward idea to train an adaptive reweighting approach is to set the weights of entities and relations as trainable parameters, and train the weights and embeddings simultaneously without bilevel optimization. This setting is not applicable in our scenario both empirically and theoretically. We detail the reason as follows.

(i) Empirically, we find this straightforward setting cannot achieve satisfactory performance. Indeed, this is in line with previous studies [4, 28], which show simultaneously learning the representations and weights for training data may hinder both learning processes.

(ii) Theoretically, many state-of-the-art KGE models utilize hinge loss or softplus loss as the objective function. Under this setting, weights cannot be trained well. Taking the widely used model TransE as an example, by taking the derivative of Eq. (1), the gradient of the weight is $\frac{\partial L_w(\Theta)}{\partial W} = L(\Theta)$. Since TransE utilizes margin-based hinge loss as the objective, the gradient of weight, $L(\Theta)$, would always be positive, thus the value of weight can only be updated along a single direction, which cannot guarantee a sufficient training for W.

To tackle the above issue, in the paper, we specially tailor bilevel optimization to learn the representations (inner level) and weights (outer level) of entities and relations. It is worth noting that we initialize the weights of entities and relations as the way Eq. (4) shows. We empirically find this initialization trick is beneficial to achieve better performance.

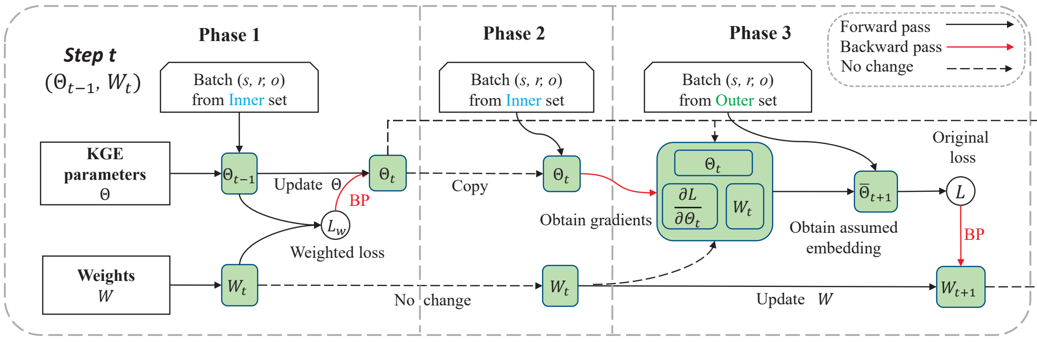


Figure 3: An illustration of the bilevel optimization process.

4.1.3 Bilevel Optimization. To learn more robust representations for entities and relations, WeightE utilizes bilevel optimization for KGE. Specifically, the inner level learns embeddings, while the outer level obtains the weights. The nested optimization problem, which is formulated as Eq. (3) is difficult to solve directly. Alternating optimization [29] reduces this problem to two simpler optimizations. In effect, we iteratively perform the following two steps:

- Θ Update. In this step, we fix W while Θ is optimized using the weighted loss L_w with triples (s, r, o) sampled from the inner set S_I.
- W Update. In this step, we fix Θ while W is optimized using the original KGE loss L with triples (s, r, o) sampled from the outer set S_O.

Training the inner level optimization is easy. We fix the weights and only update the KGE parameters. When training the outer loss, our goal is to find W that achieves the smallest loss value. In this step, the outer loss of the current model $L(\Theta_t)$, i.e., the original KGE loss, is independent of W_t, where W_t and Θ_t are the weight and KGE parameters at the t-th iteration. If Θ_{t+1} is obtained using W_t, the outer loss $L(\Theta_{t+1})$ would depend on W_t. This suggests that we can first get the next-step KGE parameters $\bar{\Theta}_{t+1}$, and then use $\bar{\Theta}_{t+1}$ and its gradient to update W_t. We use the word "assumed" because this update is never actually performed on the model we ultimately wanted. We only use it to get the direction to update W_t. We use symbols of the form $\bar{\cdot}$ to distinguish between "assumed" symbols and ordinary symbols.

4.1.4 Obtain Assumed Next-step Embedding. To update W, we need to obtain the assumed next-step embedding $\bar{\Theta}_{t+1}$. A critical step to obtain $\bar{\Theta}_{t+1}$ is to get the gradient of L_w respect to Θ_t, which is formulated as follows:

$$\frac{\partial L_w}{\partial \Theta_t} = h(W_t, \frac{\partial L}{\partial \Theta_t}) = W_t \frac{\partial L}{\partial \Theta_t}, \tag{5}$$

where the function of h is to compose the gradients. After calculating the composed gradients, the assumed next-step embedding will be obtained by $\bar{\Theta}_{t+1} = f(W_t, \frac{\partial L}{\partial \Theta_t})$, where f is the gradient update function, and will be detailed in Section 4.3.

4.1.5 Minimize the Outer Loss. So far, we have got $\bar{\Theta}_{t+1}$, and the only thing left to do is to find W that minimizes the outer loss of bilevel optimization, i.e., L. Mathematically, we want to solve

$$\arg\min_{W} \sum_{(s,r,o)\in S_O} L(\bar{\Theta}_{t+1}), \quad s.t. \; W \geq 0. \tag{6}$$

Karush-Kuhn-Tucker (KKT) conditions give feasible regions in W space for constrained minimization with non-convex objectives, which makes the search more efficient and stable. The gradient of outer loss with respect to W is denoted as G,

$$G = \nabla_W \sum_{(s,r,o)\in S_O} L(\bar{\Theta}_{t+1}). \tag{7}$$

KKT gives

$$W \cdot G = 0, \; G \geq 0, \; W \geq 0. \tag{8}$$

We can see that feasible solutions require both G and W to be non-negative with one of them equalling to zero. According to the physical meaning of W, we give a slack version which encourages W to be non-negative and G to be small.

4.1.6 Illustration. We provide an illustration of the bilevel optimization process, which is shown in Figure 3. To sum up, the whole process is divided into three phases. The first phase is the inner level, which fixes W, and updates the KGE parameters Θ. The second and the third phase correspond to the outer level optimization. Particularly, at the t-th iteration, Phase 2 gets the gradient of Θ_t with the inner set S_I. Phase 3 gets the assumed next-step embedding $\bar{\Theta}_{t+1}$, and updates the weights of entities and relations W with KKT conditions.

4.2 Application to KGE Models

In Section 4.1, we present the reweighting technique of KGE. Indeed, this technique is general and flexible, which can be applied to a number of existing KGE models. In this paper, we propose WeightE, which is built on the basis of the popular KGE model RotatE [37]. We choose RotatE due to its simplicity and efficacy. Note that we also apply the reweighting technique with other KGE models to testify the compatibility, and the results are shown in the Experiment Section (cf. Section 5.6).

We briefly introduce the base model RotatE. For a given triple (s, r, o), RotatE models each relation r as an element-wise rotation from the subject s to the object o in the complex space. Specifically, RotatE first maps the three elements s, r, and o to the complex space, then calculates a score for the triple with the rotation operation. Positive triples are supposed to have higher scores than negative ones. Given a triple (s, r, o), the score function is defined as:

$$\text{score}(v_s, v_r, v_o) = -d_r(v_s, v_o) = -\|v_s \circ v_r - v_o\|, \tag{9}$$

where v_s, v_r, and v_o are representations of s, r, and o, respectively. And $\circ$ denotes the rotation operation. $d_r(v_s, v_o)$ is a distance function, which measures the distance between $v_s \circ v_r$ and v_o. Positive triples are supposed to have smaller distances than negative ones.

The loss function of RotatE is

$$L = -\log \sigma\left(\gamma - d_r(v_s, v_o)\right) - \frac{1}{n}\sum_{i=1}^{n} \log \sigma\left(d_r\left(v_s', v_o'\right) - \gamma\right), \tag{10}$$

where γ is a fixed margin separating the positive triples with negative ones. σ is the sigmoid function. n is the negative sampling rate, i.e., we sample n negative samples for each positive triple. (s_i', r_i, o_i') is the i-th negative triple. Substituting Eq. (10) into Eq. (1), we get the weighted loss of RotatE. Substituting L and L_w into Eq. (3), we obtain the bilevel optimization of WeightE.

4.3 Optimizer Choices

WeightE can be optimized via a number of optimizers. If the model uses the SGD optimizer, the update formula to obtain $\bar{\Theta}_{t+1}$ (Phase 3 in Figure 3) is as follows:

$$\begin{aligned} \bar{\Theta}_{t+1} &= f(W_t, \frac{\partial L}{\partial \Theta_t}) \\ &= \Theta_t - \eta \frac{\partial L_w}{\partial \Theta_t} \\ &= \Theta_t - \eta W_t \frac{\partial L}{\partial \Theta_t}. \end{aligned} \tag{11}$$

To get the gradient of the outer loss L with respect to W, we also need to calculate $\frac{\partial f}{\partial W_t}$. With Eq. (11), we can obtain

$$\frac{\partial f}{\partial W_t} = -\eta \frac{\partial L}{\partial \Theta_t}, \tag{12}$$

Then with the chain rule, we can obtain the gradient of L with respect to W_t

$$\frac{\partial L}{\partial W_t} = \frac{\partial L}{\partial \bar{\Theta}_{t+1}} \frac{\partial f}{\partial W_t}. \tag{13}$$

Thanks to the deep learning frameworks such as PyTorch and TensorFlow, all the above derivation procedures can be handled automatically.

For the Adam [18] optimizer, we just need to specify f as follows:

$$\begin{aligned} \bar{\Theta}_{t+1} &= \Theta_t - \eta \frac{\sqrt{1-\beta_2^t}}{\sqrt{1-\beta_1^t}} \frac{p_t}{\sqrt{q_t} + \epsilon}, \\ p_t &= \beta_1 p_{t-1} + (1-\beta_1) \frac{\partial L_w}{\partial \Theta_t}, \\ q_t &= \beta_2 q_{t-1} + (1-\beta_2) \frac{\partial L_w}{\partial \Theta_t} \odot \frac{L_w}{\partial \Theta_t}, \end{aligned} \tag{14}$$

where β_1, β_2 represent two hyperparameters in Adam, which are set to 0.9 and 0.999. ϵ is also a hyperparameter, which is a small value to avoid the denominator being 0. $\odot$ denotes element-wise multiplication. For the other optimizers such as Adagrad [12], the optimization can also be processed in similar ways.

5 EXPERIMENT

Following a number of previous works [3, 11, 43, 46], in this section, we evaluate the learned KG embeddings on the link prediction task. Link prediction, a.k.a. knowledge graph completion, aims to predict the missing values in incomplete knowledge triples. More formally, the goal of link prediction is to predict either the subject in a given query $(?, r, o)$ or the object in a given query $(s, r, ?)$. We conduct experiments to answer the following questions:

- **RQ 1**: Does WeightE perform better than other state-of-the-art KGE models?
- **RQ 2**: How does WeightE learn the weights for entities and relations in KG?
- **RQ 3**: Can the reweighting technique be applied to other KGE models?

5.1 Datasets

We compare WeightE with state-of-the-art baselines using three benchmark datasets: FB15k-237 [38], WN18RR [11], and YAGO3-10 [11]. FB15k-237 is extracted from Freebase [2], which provides general facts of the world. WN18RR is obtained from WordNet [22], which provides semantic knowledge of words. YAGO3-10 is a subset of YAGO3 [21] that describes the attributes of persons. The three datasets are popular benchmarks for KGE, and are widely used by a number of previous works. The statistics of the datasets are summarized in Table 1. And the statistics of frequency-aware triples in each datasets are summarized in Table 2.

Table 1: Dataset statistics.

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

Table 2: Statistics of frequency-aware triples.

Dataset	#Train	#Frequecy-aware Triples	Proportion
FB15k-237	272,115	93645	34.41%
WN18RR	86,835	21880	25.20%
YAGO3-10	1,079,040	391773	36.31%

5.2 Baselines

In this paper, we compare the proposed method with the following baselines.

- Geometric methods, including TransE [3], RotatE [37], HAKE [53] and MuRMP [42].
- Tensor decomposition methods, including DistMult [46], Complex [39], TuckER [1] and MEI [23].
- Neural network methods, including ConvE [11], ConvKB [24], CompGCN [40] and MRGAT [9].

Besides, we also compare WeightE with two previous works that focus on the data imbalance issue.

- TranSparse [16], a KGE model that alleviates the data imbalance issue with sparse mapping matrices.
- LSU [55], a recent method that tackles the data imbalance issue with latent semantic units.

In addition, we also compare WeightE with three variants.

- WeightE-param, which abandons the bilevel training procedure. For WeightE-param, we set the weights of entities and relations as trainable parameters, and train the weights and embeddings of entities and relations simultaneously.
- WeightE-fixed, which indicates that we fix the weights of entities and relations during training. Specifically, the weights are set up with Eq. (4).
- WeightE-one, which denotes we initialize the weights of entities and relations with 1, regardless of their frequency. Since WeightE initializes the weights using the way Eq. (4) shows, we set this baseline to compare the effectiveness of different initialization methods.

5.3 Experimental Settings

In the training stage, we adopt Adam [18] as the optimizer to update all the parameters. All the hyper-parameters are decided based on the model performance over the validation set via grid search. The search space is decided as follows: embedding size $\{125, 250, 500, 1000\}$, batch size $\{128, 256, 512, 1024\}$, fixed margin $\{1, 5, 10, 15, 20\}$. During the training process, before starting each epoch, we randomly split 80% of training set as the inner set S_I, and 20% as outer set S_O. We clip the weights of entities and relations between $[0.5, 20]$, this trick prevents over- or under-weighting during the training process.

In the test stage, we replace the subject and the object with all entities in KG in turn for each triple in the test set. Then we compute a score for each corrupted triple, and rank all the candidate entities according to the scores. Particularly, positive candidates are supposed to precede negative ones. Finally, we collect the rank of the correct entity. Two metrics are used to compare the performance of WeightE with other competitors: (i) Mean Reciprocal Rank (MRR, the mean of all the reciprocals of predicted ranks); (ii) Hits@n (H@n, the proportion of ranks not larger than n). All the results are reported in the "filtered" setting [3].

5.4 Overall Performance (RQ1)

Experimental results are shown in Table 3. From Table 3, we have the following findings. (1) Generally, WeightE achieves the best results compared with all the baselines, which indicates the superiority of the proposed method. From Table 2, the frequency-aware triples, in front of which we apply an adaptive weight, only account for 34.41%, 25.20%, and 36.31% of the training data in FB15k-237, WN18RR, and YAGO3-10, respectively. Yet WeightE is able to achieve substantial improvements against competitors. We further analyze the reason. If we name the entities/relations contained by frequency-aware triples as seed entities/relations, then for the three datasets, triples containing at least one seed entity/relation account for 99.4%, 100%, and 100% of the training set, respectively, i.e., almost all triples. Seed entities/relations can further influence the training of other entities and relations in the same triple, thus passing the profitable information from the reweighting technique to all the entities and relations, which guarantees the effectiveness of WeightE. (2) WeightE substantially achieves better performance than the base model RotatE. It is worth noting that for a fair comparison with other baselines, WeightE does not use the self-adversarial negative sampling trick [37] in RotatE, while WeightE still consistently outperforms RotatE, which shows the reweighting technique is of great value for KGE. (3) WeightE-param does not achieve satisfactory performance. This result is in line with previous studies [4, 28] that when the number of samples is large, simultaneously learning the weights and representations of samples may harm both training processes, leading to unsatisfactory performance. Also, as stated in Section 4.1.2, a fixed gradient direction hinders the training of weights. (4) WeightE achieves better results than WeightE-fixed, which shows that the adaptive learning process of weights is crucial for model training. And the technique of bilevel optimization, which we specially tailor for KGE in this paper, is able to learn appropriate weights for entities and relations. (5) WeightE-one outperforms all the baselines on 10 out of 12 metrics, which shows even without the frequency-aware initialization, our reweighting technique is of great benefit for KGE. We also find WeightE obtains better results than WeightE-one, which indicates the frequency-aware initialization is capable of further improving the results.

5.5 Weight Analysis (RQ2)

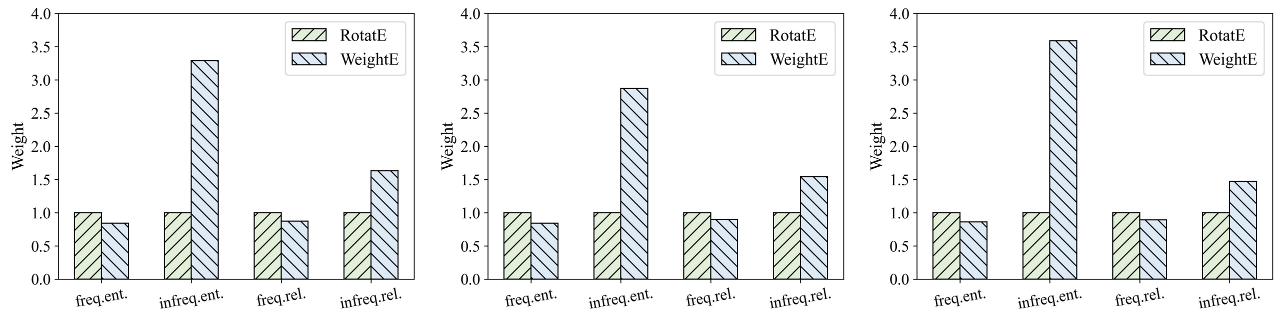
In this section, we provide further analysis about the weights learned by our reweighting technique.

5.5.1 Comparison between the weights in WeightE and RotatE. We provide comparison between the weights in the proposed model WeightE and the base model RotatE. The results are shown in Figure 4. Since RotatE does not learn weights for entities and relations, all weights are fixed as 1 in RotatE. Comparing weights in WeightE and the base model RotatE, we find that for infrequent entities and relations, the average weights in WeightE are higher than that in RotatE. While for frequent entities and relations, average weights in WeightE are slightly lower than that in RotatE. Taking the FB15k-237 dataset as an example, the average weights for frequent and infrequent entities are 0.84 and 3.29, respectively. The result is in line with our expectations, i.e., lift the weights of infrequent entities/relations, and lower the weights of high-frequency ones. This observation indicates that the proposed reweighting technique is capable of learning appropriate weights for entities and relations.

5.5.2 Effect of the reweighting technique on different entities and relations. To further analyze the proposed reweighting technique on frequent and infrequent entities and relations, we divide the test set of each dataset into two categories: (1) triples that contain at least one infrequent element, i.e., for a (s, r, o) triple, at least one of s, r, and o is an infrequent item; (2) other triples, i.e., for a (s, r, o) triple, s, r, and o are all frequent items. These two categories are denoted as infrequent and frequent triples in Figure 5, respectively. From Figure 5, we have the following findings. (1) WeightE outperforms the base model RotatE on both categories of triples, which indicates the proposed technique is able to learn better representations for both frequent and infrequent entities

Table 3: Evaluation results on FB15k-237, WN18RR and YAGO3-10 datasets. Superscripts †, ‡, ♯, § indicate the results are taken from [53], [42], [30] and the original paper, respectively. ◇ indicates the results are obtained by ourselves. ∗ denotes the improvement of WeightE is statistically significant compared with the best baseline at p-value < 0.05 over paired t-test.

	FB15k-237				WN18RR				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE†	0.294	-	-	0.465	0.226	-	-	0.501	-	-	-	-
RotatE‡	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571	0.495	0.402	0.550	0.670
HAKE†	0.346	0.250	0.381	0.542	0.497	**0.452**	0.516	0.582	0.545	0.462	0.596	0.694
MuRMP‡	0.358	0.273	0.394	**0.561**	0.481	0.441	0.496	0.569	0.495	0.448	0.591	0.698
DistMult‡	0.241	0.155	0.263	0.419	0.430	0.390	0.440	0.490	0.340	0.240	0.380	0.540
ComplEx‡	0.247	0.158	0.275	0.428	0.440	0.410	0.460	0.510	0.360	0.260	0.400	0.550
TuckER§	0.358	0.266	0.394	0.544	0.470	0.443	0.482	0.526	-	-	-	-
MEI§	0.359	0.266	0.395	0.544	0.458	0.426	0.470	0.521	-	-	-	-
ConvE‡	0.325	0.237	0.356	0.501	0.430	0.400	0.440	0.520	0.440	0.350	0.490	0.620
ConvKB$^\sharp$	0.230	0.140	-	0.415	0.249	0.056	-	0.525	0.420	0.322	-	0.605
CompGCN‡	0.355	0.264	0.390	0.535	0.479	0.443	0.494	0.546	0.489	0.395	0.500	0.582
MRGAT§	0.358	0.266	0.386	0.542	0.481	0.443	0.501	0.568	0.552	0.439	0.561	0.698
TranSparse$^\diamond$	0.344	0.233	0.384	0.529	0.488	0.428	0.493	0.569	0.491	0.396	0.539	0.661
LSU$^\diamond$	0.336	0.251	0.364	0.508	0.475	0.402	0.468	0.499	0.484	0.409	0.511	0.653
WeightE-param	0.235	0.142	0.251	0.413	0.354	0.317	0.395	0.466	0.373	0.288	0.454	0.521
WeightE-fixed	0.353	0.251	0.379	0.546	0.483	0.435	0.499	0.572	0.529	0.453	0.596	0.684
WeightE-one	0.367	0.274	0.398	0.554	0.499	0.448	0.519	0.590	0.579	0.501	0.625	0.711
WeightE	**0.371***	**0.281***	**0.404***	0.557	**0.501***	0.448	**0.520***	**0.592***	**0.580***	**0.504***	**0.628***	**0.713***



(a) FB15k-237 (b) WN18RR (c) YAGO3-10

Figure 4: Average weights for frequent/infrequent entities/relations in WeightE and RotatE. "freq." and "infreq." are short for frequent and infrequent, respectively. "ent." and "rel." are short for entity and relation, respectively.

and relations. (2) The improvement of MRR score on infrequent triples is larger than that on frequent ones. Taking the YAGO3-10 dataset as an example, the MRR improvement on infrequent triples is 0.107, while the improvement on frequent triples is only 0.040. This results indicates our method is particularly good at improving the representations of low-frequency entities and relations. This finding is in line with our expectations, i.e., alleviating the effect of long-tail distribution on KGE models.

5.5.3 Case studies. We also provide case studies to see the effect of the proposed reweighting technique. We randomly sample 6 test triples in the test set of FB15k-237, which are shown in Table 4. For the each triple, we collect the ranks for predicting $(s, r, ?)$ and $(?, r, o)$, and report the averaged rank. From Table 4, we have two observations. (1) The ranks after reweighting are significantly improved compared with ranks before reweighting, which again indicates the superiority of the proposed reweighting technique. (2) We find the weights of high-frequency entities/relations is below 1.0, while the weights of low-frequency ones are above 1.0. This observation validates that WeightE is able to assign appropriate weights to entities/relations, i.e., lift the weights for low-frequency ones, while lower the weights for high-frequency ones.

Table 4: Case Studies for triples in FB15k-237. Red and blue fonts denote high-frequency and low-frequency entities/relations, respectively. Numbers in square brackets are learned weights.

	Triples	Rank Before Reweighting	Rank After Reweighting
1	(A Beautiful Mind [0.79], film_language [0.87], English [0.52])	7	2
	(Denzel Washington [0.73], person_nationality [0.73], United States of America [0.5])	8	1.5
2	(Apple Inc. [1.24], place_founded [3.62], Cupertino [1.29])	10	2.5
	(Transformers [8.73], film_written_by [1.97], Roberto Orci [1.28])	14	3
3	(A.I. Artificial Intelligence [1.06], film_produced_by [1.12], Steven Spielberg [0.67])	5	1
	(Florida [0.76], location_contains [0.72], Tallahassee [1.31])	7	2.5

Table 5: Results of reweighting upon different representative KGE models.

	FB15k-237				WN18RR				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE	0.294	-	-	0.465	0.226	-	-	0.501	-	-	-	-
TransE + Reweighting	**0.343**	**0.247**	**0.351**	**0.499**	**0.287**	**0.265**	**0.307**	**0.529**	**0.508**	**0.419**	**0.512**	**0.677**
DistMult	0.241	0.155	0.263	0.419	0.430	0.390	0.440	0.490	0.340	0.240	0.380	0.540
DistMult + Reweighting	**0.292**	**0.223**	**0.336**	**0.483**	**0.475**	**0.423**	**0.466**	**0.507**	**0.376**	**0.282**	**0.413**	**0.579**
ConvKB	0.230	0.140	-	0.415	0.249	0.056	-	0.525	0.420	0.322	-	0.605
ConvKB + Reweighting	**0.276**	**0.203**	**0.322**	**0.481**	**0.297**	**0.192**	**0.311**	**0.543**	**0.479**	**0.381**	**0.524**	**0.663**
RotatE	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571	0.495	0.402	0.550	0.670
WeightE	**0.371**	**0.281**	**0.404**	**0.557**	**0.501**	**0.448**	**0.520**	**0.592**	**0.580**	**0.504**	**0.628**	**0.713**

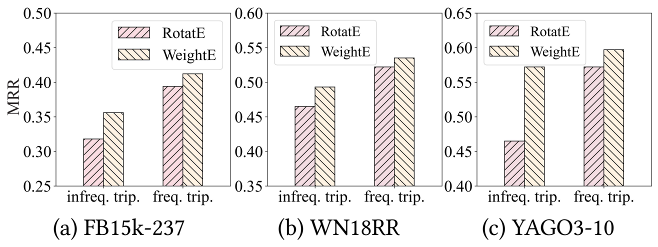


Figure 5: Average MRR score of different test triples. "infreq.", "freq.", and "trip." are short for infrequent, frequent, and triples, respectively.

5.6 Model Flexibility (RQ3)

To testify the flexibility of the reweighting technique, we apply the proposed technique to other representative KGE models, including TransE, DistMult, and ConvKB. Note that the above three models are representative geometric method, tensor factorization method, and neural network method, respectively. The results are shown in Table 5. We divide the results into four groups, and each group contains the base model and its reweighting counterpart. Results in bold font indicate the better results in each group. We find the reweighting versions consistently and substantially outperform the original versions, which manifests the flexibility and extendibility of the proposed technique. Also, the results demonstrate that the technique is compatible with a wide range of KGE models, and could serve as a plug-and-play component for a number of existing methods. We leave the potential of the reweighting technique upon more advanced KGE models to be discovered.

6 CONCLUSION

In this paper, we propose a novel KGE method WeightE, which attends differentially to different entities and relations. Particularly, WeightE specially tailors bilevel optimization for the KGE task, where the inner loop attempts to learn reliable entity and relation representations, and the outer loop aims to assign appropriate weights for each entity and relation. With the reweighting technique, WeightE is able to endow higher weights to low-frequency entities and relations, and lower weights to high-frequency ones. Furthermore, the proposed reweighting technique is general and flexible, which can be applied to a number of existing KGE models. Evaluation on three benchmark datasets demonstrates the effectiveness of WeightE.

ACKNOWLEDGEMENTS

The research work is supported by the National Key Research and Development Program of China under Grant No. 2021ZD0113602, the National Natural Science Foundation of China under Grant No. 62176014 and 62206266, and the Fundamental Research Funds for the Central Universities. Zhao Zhang is supported by the China Postdoctoral Science Foundation under Grant No. 2021M703273.

REFERENCES

[1] Ivana Balažević, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 5185–5194.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[3] 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 (2013).

[4] Zalán Borsos, Mojmir Mutny, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems* 33 (2020), 14879–14890.

[5] Mingyang Chen, Wen Zhang, Wei Zhang, Qiang Chen, and Huajun Chen. 2019. Meta Relational Learning for Few-Shot Link Prediction in Knowledge Graphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 4217–4226.

[6] Xiang Chen, Yue Cao, and Xiaojun Wan. 2021. WIND: Weighting Instances Differentially for Model-Agnostic Domain Adaptation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2366–2376.

[7] Yihong Chen, Bei Chen, Xiangnan He, Chen Gao, Yong Li, Jian-Guang Lou, and Yue Wang. 2019. λopt: Learn to regularize recommender models in finer levels. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 978–986.

[8] Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research* 153, 1 (2007), 235–256.

[9] Guoquan Dai, Xizhao Wang, Xiaoying Zou, Chao Liu, and Si Cen. 2022. MRGAT: Multi-Relational Graph Attention Network for knowledge graph completion. *Neural Networks* 154 (2022), 234–245.

[10] Songgaojun Deng, Huzefa Rangwala, and Yue Ning. 2020. Dynamic knowledge graph based multi-event forecasting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1585–1595.

[11] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

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

[13] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. 2018. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*. PMLR, 1568–1577.

[14] Chang Gao, Chengjie Sun, Lili Shan, Lei Lin, and Mingjiang Wang. 2020. Rotate3D: Representing Relations as Rotations in Three-Dimensional Space for Knowledge Graph Embedding. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 385–394.

[15] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[16] Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Thirtieth AAAI conference on artificial intelligence*.

[17] Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 792–802.

[18] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[19] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *Proceedings of the AAAI conference on artificial intelligence*. 2181–2187.

[20] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2168–2178.

[21] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference.

[22] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[23] Hung Nghiep Tran and Atsuhiro Takasu. 2020. Multi-Partition Embedding Interaction with Block Term Format for Knowledge Graph Completion. In *24th European Conference on Artificial Intelligence*. IOS Press, 833–840.

[24] Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 327–333.

[25] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[26] Guanglin Niu, Yang Li, Chengguang Tang, Ruiying Geng, Jian Dai, Qiao Liu, Hao Wang, Jian Sun, Fei Huang, and Luo Si. 2021. Relational learning with gated and attentive neighbor aggregator for few-shot knowledge graph completion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 213–222.

[27] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. Lego: Latent execution-guided reasoning for multi-hop question answering on knowledge graphs. In *International Conference on Machine Learning*. PMLR, 8959–8970.

[28] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *International conference on machine learning*. PMLR, 4334–4343.

[29] Steffen Rendle. 2012. Learning recommender systems with adaptive regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*. 133–142.

[30] Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 2 (2021), 1–49.

[31] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. 593–607.

[32] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. 2008. Resampling or reweighting: A comparison of boosting implementations. In *2008 20th IEEE International Conference on Tools with Artificial Intelligence*, Vol. 1. IEEE, 445–451.

[33] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2017. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation* 22, 2 (2017), 276–295.

[34] Ying Su, Hongming Zhang, Yangqiu Song, and Tong Zhang. 2022. Rare and Zero-shot Word Sense Disambiguation using Z-Reweighting. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 4713–4723.

[35] Zhan Su, Zhicheng Dou, Yutao Zhu, and Ji-Rong Wen. 2022. Knowledge Enhanced Search Result Diversification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1687–1695.

[36] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. 697–706.

[37] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

[38] 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*. 57–66.

[39] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *The International Conference on Machine Learning*. 2071–2080.

[40] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based Multi-Relational Graph Convolutional Networks. In *International Conference on Learning Representations*.

[41] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[42] Shen Wang, Xiaokai Wei, Cicero Nogueira Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew Arnold, Bing Xiang, Philip S Yu, and Isabel F Cruz. 2021. Mixed-curvature multi-relational graph neural network for knowledge graph completion. In *Proceedings of the Web Conference 2021*. 1761–1771.

[43] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*. 1112–1119.

[44] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 564–573.

[45] Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2018. One-Shot Relational Learning for Knowledge Graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1980–1990.

[46] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

[47] Junjie Yang, Kaiyi Ji, and Yingbin Liang. 2021. Provably faster algorithms for bilevel optimization. *Advances in Neural Information Processing Systems* 34 (2021), 13670–13682.

[48] Xiangxiang Zeng, Xinqi Tu, Yuansheng Liu, Xiangzheng Fu, and Yansen Su. 2022. Toward better drug discovery with knowledge graph. *Current opinion in structural biology* 72 (2022), 114–126.

[49] Chuxu Zhang, Huaxiu Yao, Chao Huang, Meng Jiang, Zhenhui Li, and Nitesh V Chawla. 2020. Few-shot knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3041–3048.

[50] Fuwei Zhang, Zhao Zhang, Xiang Ao, Dehong Gao, Fuzhen Zhuang, Yi Wei, and Qing He. 2022. Mind the Gap: Cross-Lingual Information Retrieval with Hierarchical Knowledge Enhancement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4345–4353.

[51] Ningyu Zhang, Shumin Deng, Zhanlin Sun, Jiaoyan Chen, Wei Zhang, and Huajun Chen. 2020. Relation adversarial network for low resource knowledge graph completion. In *Proceedings of The Web Conference 2020*. 1–12.

[52] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. 2020. AutoSF: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 433–444.

[53] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3065–3072.

[54] Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. 2018. Knowledge graph embedding with hierarchical relation structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 3198–3207.

[55] Zhao Zhang, Fuzhen Zhuang, Meng Qu, Zheng-Yu Niu, Hui Xiong, and Qing He. 2021. Knowledge graph embedding with shared latent semantic units. *Neural Networks* 139 (2021), 140–148.

[56] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Chao Li, Hui Xiong, Qing He, and Yongjun Xu. 2021. Towards Robust Knowledge Graph Embedding via Multi-task Reinforcement Learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[57] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. 2020. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9612–9619.

[58] Jinfeng Zhou, Bo Wang, Ruifang He, and Yuexian Hou. 2021. CRFR: Improving conversational recommender systems via flexible fragments reasoning on knowledge graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 4324–4334.