Entity-Relation Distribution-aware Negative Sampling for Knowledge Graph Embedding

Naimeng Yao[1][0000−0003−3541−9031], Qing Liu[2][0000−0001−7895−9551], Yi Yang[3][0000−0002−8917−2196], Weihua Li[4][0000−0001−9215−4979], and Quan Bai[1][0000−0003−1214−6317]

[1] University of Tasmania, Hobart, Australia
{naimeng.yao,quan.bai}@utas.edu.au
[2] Data61, CSIRO, Hobart, Australia
q.liu@data61.csiro.au
[3] Hefei University of Technology, Hefei, China
yyang@hfut.edu.cn
[4] Auckland University of Technology, Auckland, New Zealand
weihua.li@aut.ac.nz

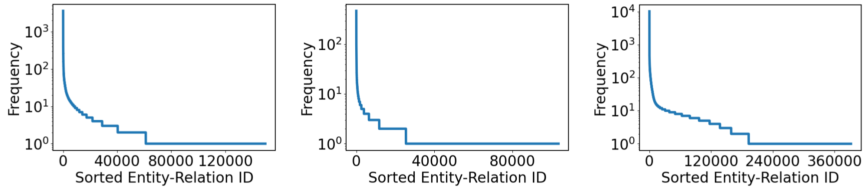
Abstract. Knowledge Graph Embedding (KGE) is a powerful technique for mining knowledge from knowledge graphs. Negative sampling plays a critical role in KGE training and significantly impacts the performance of KGE models. Negative sampling methods typically preserve a pair of **Entity-Relation (ER)** in each positive triple and replace the other entity with negative entities selected randomly from the entity set to create a consistent number of negative samples. However, the distribution of ER pairs is often long-tailed, making it problematic to assign the same number of negative samples to each ER pair, which is overlooked in most related works. This paper investigates the impact of assigning the same number of negative samples to ER pairs during training and demonstrates that this approach impedes the training from reaching the optimal solution in the negative sampling loss function and undermines the objective of the trained model. To address this issue, we propose a novel ER distribution-aware negative sampling method that can adaptively assign a varying number of negative samples to each ER pair based on its distribution characteristics. Furthermore, our proposed method also mitigates the issue of introducing false negative samples commonly found in many negative sampling methods. Our approach is founded on theoretical analysis and practical considerations and can be applied to most KGE models. We validate the effectiveness of our proposed method by testing it on conventional KGE and Neural Network-based KGE models. Our experimental results outperform most state-of-the-art negative sampling methods.

Keywords: Knowledge Graph Embedding · Negative Sampling · Knowledge Graph.

1 Introduction

A Knowledge Graph (KG) is a graph composed of entities as nodes and relations as edges that connect entities [27,25]. It is typically represented in the form of triples, denoted as $(h, r, t) \in \mathcal{F}$, where $\mathcal{F}$ is a set of facts, h and t are head and tail entities from an entity set $\mathcal{E}$, and r is drawn from a relation set $\mathcal{R}$. Knowledge Graph Embedding (KGE) represents entities and relations as vectors or matrices, which can be used to perform downstream tasks such as KG completion, relation extraction, and question answering [27,32,33]. KGE has shown promising results in various KG-related applications.

In KGE training, the scoring function, loss function, and negative sampling are crucial components. The scoring function models entity relation interactions and evaluate the likelihood of a triple's truth. The loss function processes positive and negative triple embeddings during training, assigning higher scores (from the scoring function) to positive triples and lower scores to negative ones. Simple negative sampling methods like uniform sampling [2] generate negative samples by randomly substituting t or h in each positive triple (h, r, t). The preserved Entity-Relation pair, denoted as (e, r), can represent both **Head-Relation (HR)** and **Tail-Relation (TR)** pairs, which are denoted as (h, r) and (t, r). The replaced entity is denoted as q, and the generated negative entities in negative samples are denoted as $\bar{q}$.



(a) FB15K237 (b) WN18RR (c) YAGO3-10

Fig. 1. The long-tailed distribution of ER pairs in FB15K237, WN18RR, and YAGO3-10 datasets. The y-axis represents the frequency $\#(e, r)$ of each (e, r) pair, and the x-axis shows the sorted ER ID.

Current negative sampling approaches, such as works [3,26,34], create quality negative samples that KGE models can hardly differentiate from positive triples, while others, like those mentioned in [21,1,10], demand a substantial number of negative samples for training. However, these methods typically generate the same number of negative samples for all triples, ignoring that the distribution of ER pairs is long-tailed in most datasets (Fig. 1), with a large proportion of ER pairs appearing less frequently in triples, while a few ER pairs have a higher frequency of occurrence. Recent studies have shown that imbalanced data distribution, also known as long-tailed distribution, poses a challenge to accurate model learning [32,4]. The imbalanced distribution also challenges the generation of negative samples in KGE models. Intuitively, it is unreasonable to generate the same number of negative entities $\bar{q}$ for the triple (*messi*, *profession*, q) with infrequent HR pair (*messi*, *profession*) and the triple (q, *profession*, *footballer*) with frequent and common TR pair (*footballer*, *profession*). A small number may not be sufficient for the former due to its rarity, while a large number may introduce false negative samples (i.e., true but non-observed triples) for the latter. We discover that this hinders KGE models from reaching their optimal solution, subsequently affecting their performances on downstream tasks. The main contributions of this work are as follows:

- We investigated the problem of assigning the same number of negative samples N to ER pairs for KGE training given a long-tailed distribution in KGs from a theoretical standpoint.
- We proposed an ER distribution-aware negative sampling method that is effective for a broad range of KGE models, including both conventional and Neural Network-based models.
- We conducted comprehensive experiments on six different KGE models and three benchmark KGs to validate our proposed method. The results show that our method outperforms state-of-the-art negative sampling methods.

2 Related Work

This section begins with an overview of KGE models, followed by a discussion of negative sampling methods.

2.1 Knowledge Graph Embedding Models

KGE models can be classified into two categories: conventional and Neural Network (NN)-based models. In KGE models, the scoring function calculates the plausibility score for a triple. Conventional models can be further subcategorized according to their scoring functions into the translational distance (TD) and semantic matching (SM) models. Examples of TD models include TransE [2], TransD [8], TransR [15], and RotatE [21], while popular SM models include DistMult [30], ComplEx [23], and SimplE [11]. NN-based KGE models can be broadly categorized into Convolutional NN-based (CNN) and Graph NN-based (GNN) models. Examples of CNN-based models are ConvKB [18], ConvE [5], and InteractE [24], while CompGCN [5] is a generalized version of several existing GNN methods [19,20]. GNN models usually use scoring functions from conventional KGE models as decoders for the link prediction task.

2.2 Negative Sampling Methods

The uniform sampling method selects negative entities $\bar{q}$ for (e, r) with equal probability. To avoid zero gradients in the effective training process, several negative sampling methods prioritize selecting a smaller set of quality negative samples with higher scores [3,26,34], or assigning greater weights to high score samples within multiple negative samples [21,1], which share a similar idea to train word embeddings by choosing negative words based on ranking scores in natural language processing (NLP) [7]. KBGAN [3], IGAN [26], and NSCaching [34] create a negative candidate set $C_{(e,r)}$ for (e, r) in each positive triple and employ their specific distribution functions $p_\phi(\bar{q}_i|(e,r))$ to choose negative samples. Despite minor differences in their distribution functions, their negative sampling distribution $p_n\big(\bar{q}_i|(e,r)\big)$ could be collectively represented as:

$$p_n\big(\bar{q}_i|(e,r)\big) = p_\phi\big(\bar{q}_i|(e,r)\big) = \frac{exp\Big(\alpha f\big(\bar{q}_i,(e,r)\big)\Big)}{\sum_{\big(\bar{q}_i,(e,r)\big)\in C_{(e,r)}} exp\Big(\alpha f\big(\bar{q}_i,(e,r)\big)\Big)}, \big(\bar{q}_i,(e,r)\big) \notin \mathcal{F}, \tag{1}$$

where $f(\cdot)$ is the scoring function of KGE models and α is the sampling temperature. Likewise, the distribution function serves as weights assigned to negative samples in the loss function in Self-Adv [21] and SANS [1]. However, these approaches primarily stem from experimental observations and frequently face false negative samples. Distinguishing between quality and false negative samples is difficult, as both exhibit high scores. Also, multiple negative samples can inadvertently introduce false negative samples.

Other works have analyzed the impact of the number of negative samples on the performance of KGEs. For instance, Bernoulli sampling [29] assigns different probabilities to replace the head or tail entity in a triple to construct negative samples, reducing the number of false negative samples. Trouillon et al. [23] find that increasing the number of negative samples within a certain range can improve the KGE performance. More recently, a Subsampling work [10] explores the hyperparameter tuning for the negative sampling (NS) loss function, and a Non-Sampling work [14] adds all of the negative instances in the KG for model learning.

Previous studies have not investigated the effect of assigning the same number N of negative samples to ER pairs with an imbalanced distribution, nor have they tackled the challenge of producing quality negative samples while minimizing false negative samples. To our knowledge, this study is the first to analyze the effect and attempt to resolve this issue while simultaneously mitigating false negative samples.

3 Theoretical Analysis

This section introduces the NS loss function and its optimal solution, discusses the problem of allocating an equal number of negative samples to all ER pairs, and presents our proposed solution.

3.1 Optimal Solution for the Negative Sampling Loss Function

The NS loss function was initially introduced for word-representation learning [17] and has been widely adopted in training SM models [23] for its effectiveness. Recently, the work [21] adapted it for TD models by introducing a margin term κ. The variant of the NS loss function is defined as follows:

$$L = -\frac{1}{|\mathcal{F}|} \sum_{\big(q,(e,r)\big)\in\mathcal{F}} \left[\log\sigma\Big(g\big(q,(e,r)\big)\Big) + w \sum_{\bar{q_i}\sim p_n\big(\bar{q_i}|(e,r)\big)}^{N} \log\sigma\Big(-g\big(\bar{q_i},(e,r)\big)\Big)\right], \quad (2)$$

where N represents the number of negative samples for each positive triple, $\sigma(\cdot)$ denotes the sigmoid activation function, and $g\big(q,(e,r)\big) = f\big(q,(e,r)\big) + \kappa$. In TD models, the value of κ is greater than 0, whereas it is set to 0 for SM models. When $w = p_\phi(\bar{q}|(e,r))$ (Eq. (1)), it aligns with the loss function employed in the Self-Adv method [21]. The original NS loss function corresponds to the case with $w = 1$. Recent studies [13,31,9,10] suggest that within the NS loss function ($w = 1$), the optimal embedding, denoted by $(q,(e,r))^*$ for each triple, satisfies the following condition:

$$f\big(q,(e,r)\big)^* = \log \frac{p_d\big(q|(e,r)\big)}{N\exp(\kappa)p_n\big(q|(e,r)\big)}, \quad (3)$$

where $p_d(q,(e,r))$ is the true distribution of all observed triples $(q,(e,r))$ in $\mathcal{F}$. Moreover, while $(q,(e,r))^*$ denotes the optimal embedding for a given triple, $f(q,(e,r))^*$ stands for its corresponding optimal solution in KGE models. The optimal solution guarantees the **monotonicity** property (objective) of the trained model. In the context of uniform sampling, where $p_n(q|(e,r)) = \frac{1}{|\mathcal{E}|}$,[5] and $|\mathcal{E}|$ is the size of the entity set, given any ER pair (e,r) and two distinct triples $(q_m,(e,r))$ and $(q_n,(e,r))$, if it is established that $p_d(q_m|(e,r)) > p_d(q_n|(e,r))$, then $f(q_m,(e,r))^* > f(q_n,(e,r))^*$ can be inferred.

3.2 Inaccessible Optimal Solution

During the KGE training, negative sampling methods typically generate negative samples for each triple by randomly retaining either the TR or the HR pair with an even chance. This process can be carried out simultaneously or alternately. Consequently, the optimal solution for each triple should align with Eq. (3) from both the TR and HR perspectives. Yet, for a given triple (h,r,t) and referencing Eq. (3), the optimal solution would yield two distinct values if $p_d(h|(t,r))$ is not the same as $p_d(t|(h,r))$. Nonetheless, it is impractical for a single triple to possess dual scores simultaneously, making attaining the optimal solution infeasible. Moreover, a model optimized in this manner loses its **monotonicity** trait, leading to a decline in the performance of KGE in downstream tasks.

Indeed, the distributions of $p_d(h|(t,r))$ and $p_d(t|(h,r))$ could differ significantly in most triples. Even though obtaining the explicit values of $p_d(h|(t,r))$ and $p_d(t|(h,r))$ is not feasible due to incomplete knowledge graphs, their distributions can be implicitly inferred from the dataset. For a simplified explanation,

[5] In the implementation, observed positive entities are filtered out. However, since the total number of entities $|\mathcal{E}|$ is significantly larger than the frequency $\#(e,r)$ of ER pairs, for the sake of simplification in theoretical analysis, we set $p_n(q|(e,r)) = \frac{1}{|\mathcal{E}|}$.

we use the frequency of ER pairs $\#(e,r)$ as a proxy for the distribution, assuming that for $(q,(e,r)) \in \mathcal{F}$, $p_d(q|(e,r)) = \frac{1}{\#(e,r)}$, and for $(q,(e,r)) \notin \mathcal{F}$, $p_d(q|(e,r)) = 0$. As shown in Fig. 1, ER pairs follow an imbalanced distribution, so $p_d(h|(t,r))$ and $p_d(t|(h,r))$ can differ significantly in most triples. This problem has been ignored in most research.

3.3 Assigning Varying $N_{(e,r)}$ to Entity-Relation Pair (e,r)

In pursuit of ensuring $f(h,(t,r))^* = f(t,(h,r))^*$ as per Eq. (3), a straightforward adjustment is modifying N to $N_{(e,r)}$ and assigning $N_{(e,r)}$ to each input (e,r) in proportion to $p_d\big(q|(e,r)\big)$, when $\exp(\kappa)$ and p_n are constant. Nevertheless, one prerequisite to consider is the persistence of Eq. (3) when $N_{(e,r)}$ is no longer a uniform, constant number N in Eq. (2). We find modifying N to $N_{(e,r)}$ will not disrupt Yang's proof process [31] for Eq. (3), so the subsequent proposition remains valid.

Proposition 1. *Within the NS loss where $N_{(e,r)}$ represents the number of negative samples for each ER pair (e,r), the optimal solution can still be formulated as follows:*

$$f\big(q,(e,r)\big)^* = \log \frac{p_d\big(q|(e,r)\big)}{N_{(e,r)} exp(\kappa) p_n\big(q|(e,r)\big)}. \tag{4}$$

The complete proof is available in the supplementary material. This optimal solution still guarantees the **monotonicity** property of the trained model. Moreover, practical realization is attainable by adhering to the proposition below, which outlines the suitable number of negative samples for the TR and HR pairs:

Proposition 2. *To guarantee $f(h,(t,r))^* = f(t,(h,r))^*$, the number of negative samples $N_{(h,r)}$ and $N_{(t,r)}$ should conform to the following equation when utilizing the frequency of ER pairs $\#(e,r)$ as a proxy for the distribution.*

$$\frac{N_{(t,r)}}{N_{(h,r)}} = \frac{p_d(h|(t,r))}{p_d(t|(h,r))} \approx \frac{\#(h,r)}{\#(t,r)} \Rightarrow N_{(t,r)}\#(t,r) = N_{(h,r)}\#(h,r). \tag{5}$$

For instance, considering the triple (*messi*, *profession*, *footballer*), the number of corresponding negative samples should adhere to $N_{(messi,profession)}\#(messi, profession) = N_{(footballer,profession)}\#(footballer, profession)$. Here, we refer to the collection of all observed HR pairs sharing the same TR pair as the Share Tail-Relation (STR) for (t,r), represented by $S_{(t,r)}$. Similarly, the Share Head-Relation (SHR) for (h,r) is denoted as $S_{(h,r)}$. By using the transitive property of equality, we present the following proposition:

Proposition 3. *For any HR pairs $(h_m,r),(h_n,r) \in S_{(t,r)}$, the numbers of negative samples $N_{(h_m,r)}$ and $N_{(h_n,r)}$ should satisfy:*

$$N_{(h_m,r)}\#(h_m,r) = N_{(t,r)}\#(t,r) = N_{(h_n,r)}\#(h_n,r). \tag{6}$$

Similar conclusions can be reached for TR pairs in $S_{(h,r)}$.

Considering HR pairs (*messi*, *profession*) and (*maradona*, *profession*) that share TR $(footballer,\ profession)$, the numbers of their negative samples also conform to $N_{(messi,profession)}\#(messi,$ *profession*$)$ $=N_{(maradona,profession)}\#(maradona,$ $profession)$. Eq. (5) and (6) provide guidelines for assigning numbers of negative sampling within a triple as well as within $S_{(t,r)}$ (or $S_{(h,r)}$). However, designing batches within each Share Entity-Relation (SER) during training can be challenging and may introduce bias, so we propose a global ER distribution-aware negative sampling method that considers ER pairs' distribution across the entire dataset rather than just within a single triple or SER.

4 Entity-Relation Distribution-aware Negative Sampling Method (ERDNS)

In this section, we apply the theoretical result to practical implementation, discuss the challenges faced during method development, and present the proposed method's details.

4.1 From Theory to Practice: Two-Step Expansion

Among ER pairs Share the Same Relation r. Through the mechanism of the transitive property of equality, the association between $N_{(e,r)}$ and the frequency $\#(e,r)$ can be generalized across all HR and TR pairs that have the same r. For example, while the HR pairs (*messi*, *profession*) and (*parreira*, *profession*) do not have a shared TR pair, they are each in STR sets $S_{(footballer,profession)}$ and $S_{(coach,profession)}$, both of which intersect at the HR pair (*maradona*, *profession*). Through the transitive property of equality, $N_{(messi,profession)}\#(messi,$ *profession*$)$ $=N_{(parreira,profession)}\#(parreira, profession)$ holds. On a broader scale, this inference is plausible for most relationships in the dataset.

Among All ER pairs. Assigning negative samples to ER pairs with the same r is still inefficient, so we propose extending Eq. (5) and (6) to apply among all ER pairs, as shown below:

$$N_{(e_m,r_m)}\#(e_m,r_m) = N_{(e_n,r_n)}\#(e_n,r_n), \tag{7}$$

where (e_m,r_m) and (e_n,r_n) represent any ER pairs in the dataset. Expanding the scope of Eq. (5) and (6) only extends the relationship without affecting its maintenance within a triple and within $S_{(t,r)}$ (or $S_{(h,r)}$). It is worth noting that the practical expansion does not affect the attainment of the optimal solution for the NS loss, nor does it compromise the **monotonicity** property of the trained model.

The concept introduced in Eq. (7) reflects that ER pairs that occur more frequently in the dataset should be given fewer negative samples during training, and those that occur less frequently should be given more negative samples. As noted earlier, in the case of research [3,26,34,21,1], a potential issue is the introduction of false negative samples. This drawback can negatively impact the convergence rate of the model, increase computational expenses, and ultimately reduce the overall performance. Our method also effectively reduces the likelihood of generating false negative samples. While Bernoulli sampling can also reduce false negative samples, our method has a broader scope and stronger theoretical support. Instead of emphasizing the average frequency of heads per tail *hpt* or tails per head *tph* for each r in Bernoulli sampling, we focus on the frequency of ER pairs. The average frequency *hpt* or *tph* can be distinctively different from $\#(e,r)$ in the dataset. Moreover, we assign different $N_{(e,r)}$ values to each (e,r) in the dataset, unlike Bernoulli sampling, which only considers ER pairs within a single triple. In the following subsection, we provide details about our proposed method ERDNS.

4.2 The Proposed Method

Previous negative sampling methods commonly assign N negative samples to each positive triple in a training mini-batch $\mathcal{F}_{batch}$, resulting in a total of $N_a = N \times |\mathcal{F}_{batch}|$ negative samples for triples in a mini-batch. Eq. (7) demonstrates that the number of negative samples $N_{(e_i,r_i)}$ for each triple $(q_i,(e_i,r_i)) \in \mathcal{F}_{batch}$, $i \in [1, |\mathcal{F}_{batch}|]$, is inversely proportional to $\#(e_i,r_i)$. We propose a new allocation strategy to address this issue. Instead of dividing the N_a samples according to the ratio of $\frac{1}{\#(e_i,r_i)}$, we treat the N_a negative samples as N_a independent statistical experiments, denoted as x_j, where $j \in [1, N_a]$. For each of these experiments, there are $|\mathcal{F}_{batch}|$ possible outcomes labeled as i. We then assign $N_{(e_i,r_i)}$ based on how frequently outcome i appears in the N_a experiments. The resulting allocation can be modeled using a multinomial distribution shown below.

$$\begin{aligned} p(x_j = i) = p(e_i,r_i) = \frac{\frac{1}{\#(e_i,r_i)}}{\sum_{i=1}^{|\mathcal{F}_{batch}|} \frac{1}{\#(e_i,r_i)}}, \sum_{i=1}^{|\mathcal{F}_{batch}|} p(e_i,r_i) = 1, \\ N_{(e_i,r_i)} = \sum_{j=1}^{|\mathcal{F}_{batch}|} I(x_j = i), \end{aligned} \tag{8}$$

where $I(\cdot)$ is the indicator function.
0 **Negative Sample Problem.** During the pre-processing of the dataset, we compute the frequency $\#(e,r)$ for both (h,r) and (t,r) pairs by traversing the dataset and indexing them by the respective (e,r). However, the frequency distribution of these pairs is often highly skewed, with the maximum value being significantly larger than the minimum value. This can cause some ER pairs to struggle to receive even a single negative sample, which in turn can severely impact the performance of KGE models. Thus, it is crucial to devise a reliable and effective ER distribution-aware negative sampling method that satisfies two key **requirements**: 1) generating a varying $N_{(e,r)}$ for each (e,r) by using Eq. (8); 2) ensuring fewer ER pairs (e,r) obtain zero negative samples. To meet the two requirements, we introduce a hyperparameter $\alpha \in (0,1]$:

$$p(e_i,r_i) = \frac{\frac{1}{\#(e_i,r_i)^{\alpha}}}{\sum_{i=1}^{|\mathcal{F}_{batch}|} \frac{1}{\#(e_i,r_i)^{\alpha}}}. \tag{9}$$

A smaller value of α can help meet the second requirement, ensuring that fewer pairs get zero negative samples. Our proposed strategy enhances the adaptability and robustness of the model, especially in handling the long-tailed distribution of ER pairs. We use Algorithm 1 to train KGE models based on the distribution-aware negative sampling method. Our proposed method centers around the generation of varying numbers of negative samples $N_{(e_i,r_i)}$ according to the probability of $p(e_i, r_i)$ (Step 5). The probability is dynamically calculated based on the inclusion of (e_i, r_i) in $\mathcal{F}_{batch}$ using Eq. (9). Once $N(e_i, r_i)$ are determined for each (e_i, r_i) in the triple, negative entities $\bar{q_i}$ are randomly chosen from the entity set. The remaining training processes adhere to standard KGE training steps. Our method follows unbiased uniform sampling and allocates fewer negative samples to high-frequency ER pairs to reduce the likelihood of generating false negative samples. Instead of assessing the scores of negative samples for each individual triple, our method evaluates the quality of all negative samples from a data augmentation perspective. While our method does not guarantee that every generated negative sample has a high score, the proposed allocation strategy can effectively address the issue caused by the long-tail nature, which is prevalent for many datasets, and improves the overall quality of negative samples.

Algorithm 1 Entity-Relation Distribution-aware Negative Sampling Method (ERDNS).

```
Input: Training set $\mathcal{F} = \{\big(q, (e, r)\big)\}$, embeddings of entities and relations, scoring
    function $f\big(q, (e, r)\big)$, and frequency $\#(e, r)$. Hyperparameters: $\alpha$, average number
    of negative samples $N$, dimension $d$, mini-batch size $b$, and training step $Q$.
Output: Trained embeddings.
 1: Initialize the embeddings.
 2: for $step = 1, \ldots, Q$ do
 3:     Sample a mini-batch $\mathcal{F}_{batch} \in \mathcal{F}$.
 4:     Calculate the probability $p(e_i, r_i)$ for each triple $\big(q_i, (e_i, r_i, )\big)$ in $\mathcal{F}_{batch}$.
 5:     Calculate the negative samples $N_{(e_i,r_i)}$ based on $p(e_i, r_i)$ in Eq. (9).
 6:     Randomly sample $N_{(e_i,r_i)}$ negative entities $\bar{q_i}$ for each triple.
 7:     Feed positive triples $\big(q_i, (e_i, r_i, )\big)$ into the NS loss function ($w = 1$).
 8:     Feed negative samples $\big(\bar{q_i}, (e_i, r_i)\big)$ with size $N_{(e_i,r_i)}$ into the NS loss function.
 9:     Update embeddings.
10: end for
```

Computational Cost. We examine computational space to store intermediate computation results and computational time of ERDNS using the DistMult model as a reference. The scoring function of this model is $f(q, (e, r)) = \langle e, r, q\rangle$, where $\langle\cdot\rangle$ is the inner product. The computational space and time of ERDNS and other state-of-the-art methods in each mini-batch are detailed in Table 1. The notation "1" indicates the space required to store the embedding of $\langle e, r\rangle$, and "N" represents the space for the embeddings of negative samples $\overline{q}$s. In the computational process, ERDNS needs to expand the embedding of $\langle e, r\rangle$ to align with the size of embeddings of $\overline{q}$s. This results in a manageable doubling of the standard cache space. The computation time of ERDNS is the same as others with the same average "N" negative samples. The computation cost of ERDNS is comparable to the existing negative sampling methods. Additionally, by applying ERDNS, there is an opportunity to reduce the number of negative samples, potentially lowering its computational cost compared to other methods. Therefore, ERDNS proves to be scalable for large-scale knowledge graphs.

Table 1. Comparison of the proposed method with the state-of-the-art regarding computational space and time in each mini-batch. N_1 and N_2 are the cache size and randomly sampled negative triples in NSCaching.

Method	NSCaching	KBGAN	unif	Bernoulli	Self-Adv	Subsampling	ERDNS
Space	$\mathcal{O}(b(N_1+N_2+1)d)$	$\mathcal{O}(b(N+1)d)$					$\mathcal{O}(b(2N)d)$
Time	$\mathcal{O}(b(N_1+N_2)d)$	$\mathcal{O}(bNd)$					

5 Experiments

In this section, we detail our experimental setup, present our method's outcomes, analyze the findings, and assess our approach's impact through ablation and extension studies.

5.1 Experimental Setting

Table 2. Statistics of the datasets.

Dataset	Entity	Relation	Train	Valid	Test
FB15K237	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

Datasets. We evaluate our proposed method on three commonly used knowledge graphs: FB15K237 [22], WN18RR [28], and YAGO3-10 [16]. The statistics for these KGs are presented in Table 2, and their ER distributions are illustrated in Fig. 1.

Models. To evaluate the efficacy of our proposed method ERDNS, we integrate it into six different models: four conventional KGE models and two NN-based models. These models are as follows: TD models (TransE and RotatE), SM models (DistMult and ComplEx), a CNN model (ConvE), and a GNN model (CompGCN). CompGCN has used TransE, DistMult, and ConvE as decoders.

Compared Methods. We compare our proposed method ERDNS with two categories of negative sampling methods.

With Single Negative Sample: These methods only generate one negative sample per positive triple.

• KBGAN [3] computes a probability distribution over candidate negative triples for each positive triple and selects a single negative sample by sampling from the distribution.
• NSCaching [34] uses caches to store and update quality negative samples for each ER during training and selects a negative sample from the cache.
With Multiple Negative Samples: These methods generate more than one negative sample per positive triple.
• Uniform Sampling [2] randomly selects a set of entities to serve as negative samples with an equal probability assigned to all entities in the KG.
• Bernoulli Sampling [29] generates negative samples for a positive triple (h, r, t) by replacing the head and tail according to the probabilities $\frac{tph}{tph+hpt}$ and $\frac{hpt}{tph+hpt}$.
• Self-Adv [21] generates multiple negative samples per positive triple and assigns higher weights to high score negative samples during training.
• Subsampling [10] is specialized for the NS loss and studied theoretically. It incorporates Self-Adv in its implementation.

Hyperparameter Settings. We utilized the Adam optimizer [12] to optimize the KGE models. To establish baseline hyperparameters for conventional models, we adopt settings from prior research [21,10]. These settings consist of the embedding dimension d, batch size b, learning rate lr, number of negative samples N, margin λ (for TD models), and regularization factor μ (for SM models). The results for uniform Sampling, Bernoulli Sampling, Self-Adv and Subsampling are obtained using the inherited hyperparameters. For KBGAN and NSCaching, we directly take the results from their works because the best settings were not provided. The hyperparameters for NSCaching (RotatE) were optimized based on the settings provided in its original paper. ConvE and CompGCN initially used a complete entity set for training, so we only used negative sampling methods with multiple samples in comparison. We also inherited the hyperparameters from their prior works. We identified two critical hyperparameters in our method: the average number of negative samples N per positive triple and α. We performed a grid search to evaluate the range of values, with N having options of $\{50, 128, 256, 512, 1024\}$ (Conventional models) and $\{1024, 2048, 4096, 8192\}$ (NN-based models), and α having options of $\{0.25, 0.5, 0.75, 1\}$.

Evaluation Setting. We evaluate the proposed negative sampling method using the link prediction task designed to predict the missing entity in a positive triple (h, r, t). To assess the performance of the method, we employ standard evaluation metrics, including Mean Reciprocal Rank (MRR), and Hits@N (N=$1, 3, 10$). The evaluation is performed with the filtered setting.

5.2 Main Results

Conventional Models. Table 3 displays the results of compared methods with conventional KGE models. ERDNS consistently outperforms other models on all three KGs, achieving the average highest MRR and Hits@N across different ranks. Negative sampling methods employing multiple samples generally perform better than those using single samples. Even though the latter often utilize

Table 3. Performance comparison on DistMult, ComplEx, TransE and RotatE.

Dataset	KGE model	DistMult				ComplEx			
	Method	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
FB15K237	KBGAN	26.7	-	-	45.3	28.2	-	-	45.4
	NSCaching	28.3	-	-	45.6	30.2	-	-	48.1
	unif	21.9	13.1	23.4	41.1	22.7	13.7	24.5	42.1
	Bernoulli	25.9	17.4	28.3	43.3	31.0	21.2	34.3	51.3
	Self-Adv	30.9	22.1	33.6	48.4	32.2	23.0	35.1	51.0
	Subsampling	29.9	21.2	32.7	47.5	32.8	23.6	36.1	51.2
	ERDNS	**35.0**	**25.7**	**38.3**	**53.7**	**35.7**	**26.2**	**39.3**	**54.7**
WN18RR	KBGAN	38.5	-	-	44.3	42.9	-	-	47.0
	NSCaching	41.3	-	-	45.5	44.6	-	-	50.9
	unif	42.7	39.0	43.7	50.7	44.8	41.5	46.4	51.7
	Bernoulli	40.0	37.3	40.7	45.2	46.0	42.4	47.8	53.0
	Self-Adv	43.9	39.4	45.2	53.8	47.1	42.8	48.9	55.7
	Subsampling	44.6	40.0	45.9	54.4	47.6	43.3	49.3	56.3
	ERDNS	**45.5**	**41.1**	**46.7**	**54.6**	**48.4**	**44.3**	**50.0**	**56.4**
YAGO3-10	NSCaching	40.3	-	-	56.6	40.5	-	-	57.8
	unif	45.4	35.7	51.0	64.5	46.4	36.6	51.9	64.8
	Bernoulli	49.0	39.4	54.2	67.2	50.3	40.9	55.5	68.0
	Self-Adv	44.8	34.7	50.4	64.3	47.3	37.4	53.2	66.1
	Subsampling	45.9	34.6	52.7	66.2	49.1	38.1	56.0	68.5
	ERDNS	**52.6**	**43.7**	**57.8**	**69.4**	**54.6**	**45.9**	**59.8**	**70.3**
Dataset	KGE model	TransE				RotatE			
	Method	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
FB15K237	KBGAN	29.4	-	-	46.7	-	-	-	-
	NSCaching	30.0	-	-	47.4	30.9	22.1	34.1	48.6
	unif	31.4	21.5	35.2	51.5	31.9	22.3	35.3	51.7
	Bernoulli	33.4	23.9	37.0	52.6	33.6	24.1	37.1	52.7
	Self-Adv	32.9	23.0	36.8	52.7	33.6	23.9	37.4	53.0
	Subsampling	33.6	24.0	37.3	52.9	34.0	24.5	37.6	53.2
	ERDNS	**34.2**	**24.8**	**37.9**	**53.0**	**34.5**	**25.2**	**37.9**	**53.7**
WN18RR	KBGAN	18.6	-	-	45.4	-	-	-	-
	NSCaching	20.0	-	-	47.8	47.1	42.9	48.9	55.3
	unif	22.4	1.5	40.1	51.9	47.1	42.8	48.6	55.5
	Bernoulli	23.2	2.2	41.4	52.2	47.5	43.1	49.1	56.0
	Self-Adv	22.3	1.3	40.1	52.8	47.6	43.1	49.5	57.3
	Subsampling	23.0	1.9	40.7	**53.7**	47.8	42.9	**49.8**	**57.4**
	ERDNS	**23.8**	**3.0**	**41.8**	52.8	**47.9**	**43.5**	49.4	56.5
YAGO3-10	NSCaching	30.7	-	-	50.7	42.3	33.8	47.1	59.7
	unif	48.2	37.4	55.1	67.3	49.8	40.2	55.2	67.7
	Bernoulli	50.3	40.6	56.3	67.9	50.2	40.8	55.8	67.9
	Self-Adv	51.2	41.5	57.6	68.3	50.8	41.8	56.5	67.6
	Subsampling	51.3	41.9	57.2	68.1	51.0	41.9	56.5	67.8
	ERDNS	**51.7**	**42.2**	**57.6**	**68.7**	**51.8**	**42.7**	**57.0**	**68.8**

quality negative samples, their performance may still be inferior to uniform sampling with multiple samples. Regarding negative sampling methods with multiple samples, Bernoulli sampling demonstrates superior performance over uniform sampling in reducing the number of false negative samples, particularly in the case of SM models rather than TD models. This indicates that false negative samples have a more substantial impact on SM models. The Subsampling model integrates Self-Adv in implementation, its enhancement through the subsampling technique is less effective than our method when compared to Self-Adv, even though our approach does not use Self-Adv.

The effectiveness of ERDNS is more pronounced on the FB15K237 and YAGO3-10 datasets compared to WN18RR, primarily due to the broader range of frequency values in the former datasets and the relatively lower percentage of

(e, r) pairs with a frequency of 1. Both of these factors contribute to increased diversity in all $\#(e, r)$, making the optimal solution more challenging to obtain. Consequently, ERDNS demonstrates greater improvement on these datasets. Despite relying solely on dynamic allocation, our method surpasses more complex sampling methods, such as candidate selection in KBGAN, caching in NSCaching, Subsampling, and Self-Adv.

Neural Network-based Models. Table 4 presents the performance of negative sampling methods with NN-based KGE models. For CompGCN, Bernoulli sampling is not applicable due to incompatibility with the framework. Interestingly, unlike conventional KGE models, Self-Adv does not consistently outperform the uniform sampling method, which might be attributed to incorporating false negative samples, adversely affecting the learning of global features in NN-based models. Nevertheless, ERDNS continues to outperform the uniform sampling method in most conditions, highlighting its effectiveness and universality. Incorporating global trainable feature matrices might diminish the impact of negative samples during training, resulting in a modest performance enhancement of ERDNS on NN-based models.

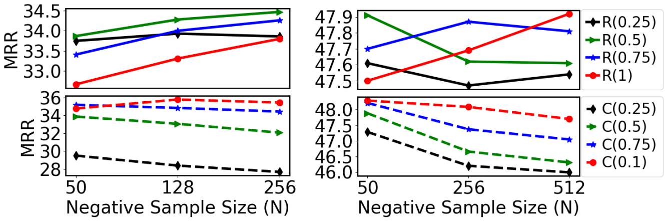
Table 4. Performance comparison on ConvE and CompGCN.

KGE model	Dataset	FB15K237				WN18RR			
	Method	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
ConvE	unif	31.5	22.6	34.4	49.8	41.0	36.9	42.4	**49.0**
	Bernoulli	29.8	21.4	32.5	46.9	39.7	35.9	41.0	47.9
	Self-Adv	31.9	22.7	34.8	**50.4**	40.3	37.0	41.4	47.0
	ERDNS	**32.3**	**23.4**	**35.4**	50.2	**41.3**	**38.0**	**42.6**	48.4
CompGCN (TransE)	unif	32.9	24.1	36.1	50.1	18.9	**5.6**	25.6	47.5
	Self-Adv	28.5	20.0	31.2	45.7	20.4	5.3	29.6	49.3
	ERDNS	**33.6**	**24.2**	**36.2**	**50.9**	**24.5**	4.9	**39.0**	**53.1**
CompGCN (DistMult)	unif	33.3	24.4	36.4	51.2	42.0	37.3	43.3	52.1
	Self-Adv	29.2	21.0	31.8	45.9	43.0	**39.2**	43.1	51.8
	ERDNS	**34.2**	**25.2**	**37.4**	**52.3**	**43.4**	**39.2**	**44.3**	**52.5**
CompGCN (ConvE)	unif	35.2	26.2	38.5	53.1	46.6	43.4	47.7	53.0
	Self-Adv	30.3	22.6	32.9	46.2	43.7	40.7	44.7	49.4
	ERDNS	**35.3**	**26.3**	**38.9**	**53.6**	**46.8**	**43.8**	**47.8**	**53.1**

5.3 Ablation Study

Investigating the Impact of α and N. As mentioned in Section 4, the parameter α is crucial in meeting the two requirements of the proposed method. A large value of α prioritizes sampling according to the distribution of ER pairs, while a small value prevents the allocation of 0 negative samples to ER pairs with high frequency. We find that N impacts the optimal value of α. When N is relatively low, setting $\alpha = 1$ may not yield the best MRR, despite being consistent with the principle, as higher values of α can lead to the "0 negative sample problem". The MRR results on RotatE (R) and ComplEx (C) for all values of α exhibit a similar trend as N increases in Fig. 2. Initially, the MRR gradually improves until it reaches its peak when N attains a sufficiently large value, denoted as N_{α}^{*}, because the "0 negative sample problem" gradually diminishes as

N increases. The sufficiently large value of N_α^* for different values of α follows the trend $N_{0.25}^* < N_{0.5}^* < N_{0.75}^* < N_1^*$. The exact value of N_α^* varies depending on the model and dataset.



(a) FB15K237

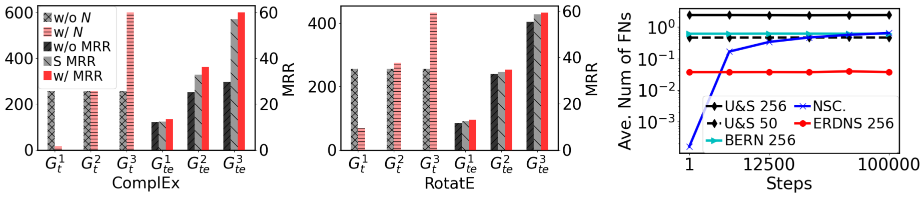
(b) WN18RR

Fig. 2. MRR of ERDNS on RotatE (R) and ComplEx (C) with different α.

Entity-Relation Pairs with Different Frequencies. We conduct experiments to examine the impact of our method on the number of negative samples $N_{(e,r)}$ for ER pairs with different frequencies during training and its effect on the performance of ER pairs with different frequencies during testing. We categorize ER pairs in the training set into three groups based on their frequencies: G_t^1 contains ER pairs with the highest frequency, where $\#(e,r) > 50$; G_t^3 contains ER pairs with the lowest frequency, where $\#(e,r) = 1$; and G_t^2 contains ER pairs with moderate frequency, where $\#(e,r) \in (1, 50]$. Additionally, we group ER pairs in the testing set into three categories, G_{te}^1, G_{te}^2, and G_{te}^3, to observe the performance of ER pairs with different frequencies. G_{te}^1 and G_{te}^2 are subsets of G_t^1 and G_t^2 respectively, while G_{te}^3 consists of ER pairs that do not appear in the training set.

Fig. 3(a) presents the average number of negative samples $N_{(e,r)}$ for G_t with and without ERDNS (w/ and w/o N), and the average link prediction results of G_{te} with and without ERDNS (w/ or w/o MRR), and Self-Adv (S MRR). With ComplEx, dynamic allocation significantly reduces the average $N_{(e,r)}$ for G_t^1 from 256 to 17, and for G_t^3, it increases from 256 to 600. The change of $N_{(e,r)}$ is slightly different for RotatE because its best result is achieved when $\alpha = 0.5$, and ComplEx performs best when $\alpha = 1$. This demonstrates that the allocation strategy significantly impacts the average $N_{(e,r)}$ in different groups. We observe that MRR of G_{te}^1, G_{te}^2, and G_{te}^3 improve with dynamic allocation, which is also better than Self-Adv, indicating that obtaining the optimal solution resulted in the overall performance improvements across ER pairs with all frequency ranges since with our method the optimal solution of all triples in the NS loss is attainable. Surprisingly, MRR of G_{te}^1 improves despite the substantial drop in average $N_{(e,r)}$ for $(e,r) \in G_{te}^1 \subset G_t^1$, suggesting that excessive negative samples may introduce false negative samples, negatively impacting the performance.

False Negative Sample. We discussed the average MRR of G_{te}^1 increases with ERDNS even though only a few negative samples are given because of introduc-



(a) Average $N_{(e,r)}$ and MRR

(b) Average FNs in G_t^1

Fig. 3. Average $N_{(e,r)}$ and MRR of three groups on different models and average numbers of false negative samples.

ing false negative samples. To demonstrate this, we present the average number of false negative samples in G_t^1 during training using various negative sampling methods such as NSCaching, uniform sampling, Bernoulli sampling, Self-Adv, and ERDNS in Fig. 3(b). It is worth noting that for NSCaching, the average number increases during training because false negative samples with high-value scores can become trapped in the caches. In the case of uniform sampling and Self-Adv (U&S), negative samples are randomly chosen from the entity set, resulting in the same average number, which is higher than that of Bernoulli sampling when N equals 256. Additionally, ERDNS shows significantly fewer false negative samples on average, almost one-hundredth of the number of uniform sampling and Self-Adv, and even much lower than them with $N = 50$.

Statistical Analysis. We conduct a paired t-test on the results obtained from 10-fold cross-validation [6] of KGE models, namely ComplEx, RotatE, and ConvE, using the FB15K237 and WN18RR datasets. Table 5 displays the average evaluation metrics from this 10-fold cross-validation and the associated paired t-test p-values. We observe that varying the train-test split percentage affects KGE model performance. Specifically, performance on the FB15K237 dataset improves, while it declines for WN18RR. However, consistent comparative results are achieved. ERDNS still exhibits the best results for SM models on FB15K237, as explained in Section 5.2. It is worth noting that the p-values consistently remain under the 0.05 threshold, which is the conventional threshold for denoting statistical significance, and frequently drop to values less than 0.0001. This suggests that even minor performance improvements exhibit statistical differences. Apart from two exceptions, our method's average results surpass those of other methods, indicating the overall ERDNS performances are better than others.

Table 5. Mean results of 10-fold cross-validation and its paired t-test p-values.

KGE model	Dataset	FB15K237				WN18RR			
	Method	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
ComplEx	Subsampling(Mean)	37.6	26.0	43.4	60.4	41.0	37.2	42.5	**48.6**
	ERDNS(Mean)	**41.8**	**30.4**	**47.8**	**64.0**	**41.3**	**37.9**	**42.7**	47.8
	p-value	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
RotatE	Subsampling(Mean)	40.1	28.3	46.8	63.0	41.7	37.5	43.1	50.2
	ERDNS(Mean)	**40.3**	**28.4**	**47.2**	**63.3**	**41.9**	**37.6**	**43.3**	**50.4**
	p-value	0.00	0.03	0.00	0.00	0.00	0.03	0.01	0.00
ConvE	Self-Adv(Mean)	38.3	26.9	43.7	**61.1**	33.9	31.5	34.8	38.8
	ERDNS(Mean)	**38.7**	**27.9**	**43.9**	59.8	**34.7**	**31.7**	**35.9**	**40.6**
	p-value	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00

6 Conclusion

In conclusion, our study reveals that the current negative sampling method ignores the problem that the optimal solution in the NS loss is not attainable when the same number of negative samples are assigned to ERs with an imbalanced distribution. To address this issue, we present an ER distribution-aware negative sampling method that generates varying numbers $N_{(e,r)}$ of negative samples for each (e, r) based on their distribution in the dataset. The method also can effectively alleviate the problem of introducing false negative samples in many negative sampling methods. The proposed method takes into account both theoretical and practical aspects and is applicable to a wide range of KGE models. Experimental results on the link prediction task demonstrate the effectiveness of the proposed method on both conventional and NN-based KGE models.

The results of NN-based models indicate that incorporating global trainable features may impact the effectiveness of negative samples in the KGE training process. In future research, we aim to investigate this issue and explore methods further to enhance the effectiveness of our approach to NN-based models.

Supplemental Material Statement: Detailed proofs, source code, scoring functions of KGE models, best hyperparameter settings, and full 10-fold cross-validation results are all available at https://github.com/for4ever44/ERDNS.

References

1. Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W.L., Bose, A.J.: Structure aware negative sampling in knowledge graphs. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6093–6101 (2020)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems **26** (2013)
3. Cai, L., Wang, W.Y.: Kbgan: Adversarial learning for knowledge graph embeddings. CoRR (2017)
4. Chen, X., Zhou, Y., Wu, D., Zhang, W., Zhou, Y., Li, B., Wang, W.: Imagine by reasoning: A reasoning-based implicit semantic data augmentation for long-tailed

classification. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 356–364 (2022)

5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
6. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation **10**(7), 1895–1923 (1998)
7. Guo, G., Ouyang, S., Yuan, F., Wang, X.: Approximating word ranking and negative sampling for word embedding. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence. p. 4092–4098 (2018)
8. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers). pp. 687–696 (2015)
9. Kamigaito, H., Hayashi, K.: Unified interpretation of softmax cross-entropy and negative sampling: With case study for knowledge graph embedding. 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). pp. 5517–5531 (2021)
10. Kamigaito, H., Hayashi, K.: Comprehensive analysis of negative sampling in knowledge graph representation learning. In: International Conference on Machine Learning. pp. 10661–10675. PMLR (2022)
11. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. Advances in neural information processing systems **31** (2018)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. Technical report (2014)
13. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. Advances in neural information processing systems **27** (2014)
14. Li, Z., Ji, J., Fu, Z., Ge, Y., Xu, S., Chen, C., Zhang, Y.: Efficient non-sampling knowledge graph embedding. In: Proceedings of the Web Conference 2021. pp. 1727–1736 (2021)
15. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 2181–2187 (2015)
16. Mahdisoltani, F., Biega, J., Suchanek, F.: Yago3: A knowledge base from multilingual wikipedias. In: 7th biennial conference on innovative data systems research. CIDR Conference (2014)
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems **26** (2013)
18. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.: 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). pp. 327–333 (2018)
19. Schlichtkrull, M., Kipf, T.N., Bloem, P., Berg, R.v.d., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European semantic web conference. pp. 593–607 (2018)

20. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3060–3067 (2019)
21. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: International Conference on Learning Representations (2019)
22. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality. pp. 57–66 (2015)
23. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
24. Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., Talukdar, P.: Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3009–3016 (2020)
25. Wang, M., Qiu, L., Wang, X.: A survey on knowledge graph embeddings for link prediction. Symmetry **13**(3), 485 (2021)
26. Wang, P., Li, S., et al.: Incorporating gan for negative sampling in knowledge representation learning. CoRR (2018)
27. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
28. Wang, Y., Ruffinelli, D., Gemulla, R., Broscheit, S., Meilicke, C.: On evaluating embedding models for knowledge base completion. Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019) (2019)
29. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI conference on artificial intelligence. vol. 28 (2014)
30. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: 3rd International Conference on Learning Representations (ICLR) (2015)
31. Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., Tang, J.: Understanding negative sampling in graph representation learning. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 1666–1676 (2020)
32. Zhang, N., Deng, S., Sun, Z., Wang, G., Chen, X., Zhang, W., Chen, H.: Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. 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). pp. 3016–3025 (2019)
33. Zhang, W., Paudel, B., Wang, L., Chen, J., Zhu, H., Zhang, W., Bernstein, A., Chen, H.: Iteratively learning embeddings and rules for knowledge graph reasoning. In: The World Wide Web Conference. pp. 2366–2377 (2019)
34. Zhang, Y., Yao, Q., Shao, Y., Chen, L.: Nscaching: simple and efficient negative sampling for knowledge graph embedding. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE). pp. 614–625 (2019)