mathematics

Article

Universal Knowledge Graph Embedding Framework Based on High-Quality Negative Sampling and Weighting

Pengfei Zhang [1], Huang Peng [1], Yang Fang [2,*], Zongqiang Yang [1], Yanli Hu [1], Zhen Tan [1] and Weidong Xiao [1]

1 Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China
2 College of Information and Communication, National University of Defense Technology, Changsha 410073, China
* Correspondence: fangyang12@nudt.edu.cn

Abstract: The traditional model training approach based on negative sampling randomly samples a portion of negative samples for training, which can easily overlook important negative samples and adversely affect the training of knowledge graph embedding models. Some researchers have explored non-sampling model training frameworks that use all unobserved triples as negative samples to improve model training performance. However, both training methods inevitably introduce false negative samples and easy-to-separate negative samples that are far from the model's decision boundary, and they do not consider the adverse effects of long-tail entities and relations during training, thus limiting the improvement of model training performance. To address this issue, we propose a universal knowledge graph embedding framework based on high-quality negative sampling and weighting, called HNSW-KGE. First, we conduct pre-training based on the NS-KGE non-sampling training framework to quickly obtain an initial set of relatively high-quality embedding vector representations for all entities and relations. Second, we design a candidate negative sample set construction strategy that samples a certain number of negative samples that are neither false negatives nor easy-to-separate negatives for all positive triples, based on the embedding vectors obtained from pre-training. This ensures the provision of high-quality negative samples for model training. Finally, we apply weighting to the loss function based on the frequency of the entities and relations appearing in the triples to mitigate the adverse effects of long-tail entities and relations on model training. Experiments conducted on benchmark datasets FB15K237 and WN18RR using various knowledge graph embedding models demonstrate that our proposed framework HNSW-KGE, based on high-quality negative sampling and weighting, achieves better training performance and exhibits versatility, making it applicable to various types of knowledge embedding models.

Keywords: knowledge graph; high-quality negative sampling; universal embedding

MSC: 05C62

check for updates

Citation: Zhang, P.; Peng, H.; Fang, Y.; Yang, Z.; Hu, Y.; Tan, Z.; Xiao, W. Universal Knowledge Graph Embedding Framework Based on High-Quality Negative Sampling and Weighting. *Mathematics* **2024**, *12*, 3489. https://doi.org/10.3390/math12223489

Academic Editor: Florin Leon

Received: 7 October 2024
Revised: 31 October 2024
Accepted: 4 November 2024
Published: 8 November 2024

1. Introduction

Currently, knowledge graphs (KGs) serve as important carriers for storing and processing structured data, and, as advanced knowledge interconnection solutions, they provide extensive intelligent system capabilities for many applications, garnering significant attention from both academia and industry. Knowledge graph embedding (KGE) involves embedding entities and relations from KGs into continuous vector spaces, enabling efficient computation of semantic information while preserving the inherent structure of the knowledge graph. This makes large-scale applications of knowledge graphs a reality. Consequently, in recent years, knowledge graph embeddings have been widely applied in scenarios such as information retrieval [1], recommendation systems [2], and question-answering systems [3].

In recent years, researchers have proposed various KGE models, with classic models including TransE [4], DistMult [5], SimplE [6], and ComplEx [7], among others. To better extract useful information from the dataset for improved knowledge embeddings, KGE models typically adopt traditional negative sampling methods for training, as shown in Figure 1. 'Neg' denotes negative samples. Initially, negative samples are sampled based on certain strategies from the positive samples in the dataset. Afterwards, the model's scoring function is used to compute the loss function for both positive and negative samples, which is optimized during training to obtain embedding representations of entities and relations in the knowledge graph. Traditional knowledge graph embedding models predominantly depend on negative sampling, which constructs a candidate negative sample set relative to the positive sample set based on certain strategies and selects a portion of negative samples for the training process. Common negative sampling strategies include random negative sampling, adversarial negative sampling, and negative sampling that incorporates additional information. However, knowledge graphs face the challenge of data sparsity. The traditional negative sampling training approach even exacerbates the data sparsity by only sampling a subset of negative samples for training, leading to insufficient training samples and potentially overlooking important negative samples, thus impacting the performance of model training.

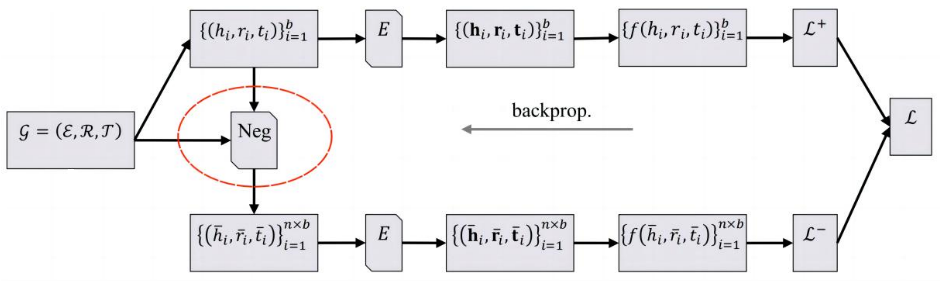
$\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$
$\{(h_i, r_i, t_i)\}_{i=1}^{b}$
E
$\{(\mathbf{h}_i, \mathbf{r}_i, \mathbf{t}_i)\}_{i=1}^{b}$
$\{f(h_i, r_i, t_i)\}_{i=1}^{b}$
$\mathcal{L}^+$
Neg
backprop.
$\mathcal{L}$
$\{(\bar{h}_i, \bar{r}_i, \bar{t}_i)\}_{i=1}^{n \times b}$
E
$\{(\bar{\mathbf{h}}_i, \bar{\mathbf{r}}_i, \bar{\mathbf{t}}_i)\}_{i=1}^{n \times b}$
$\{f(\bar{h}_i, \bar{r}_i, \bar{t}_i)\}_{i=1}^{n \times b}$
$\mathcal{L}^-$

Figure 1. Traditional negative sampling training method.

Researchers have explored non-sampling training methods, which involve using all non-positive triples from the knowledge graph—comprising all entities and relations—as negative samples for training. This approach aims to resolve the issues associated with traditional negative sampling methods by skipping the sampling process. However, this training method has several shortcomings. First, by considering all unobserved triples as negative samples, it can easily introduce false negative samples, that is, these factual triples that have not been included in the knowledge graph dataset due to data sparsity, which can affect the accuracy of model training. Second, this method introduces a large number of easy-to-separate negative samples that are far from the model's decision boundary. Studies such as those in references [8,9] have shown that most negative triples are easy-to-separate negatives, which contribute little to the gradient update direction during model training. This can negatively impact the convergence speed and accuracy of the model.

In summary, an analysis of these two traditional training methods reveals a common problem: neither ensures the participation of high-quality negative samples in training. According to reference [10], a high-quality negative sample is defined as one that is difficult to judge as true or false without referencing the true value; it is semantically meaningful but factually incorrect. In other words, high-quality negative samples are neither false negatives nor easy-to-separate negatives. Traditional negative sampling methods randomly sample unobserved samples as negative samples for training while non-sampling methods include all unobserved samples as negatives. Neither approach guarantees the quality of the negative samples used in training, thereby limiting the improvement of model training performance. As shown in Figure 2, reference [7] presents link prediction experiments of the ComplEx model on the FB15K dataset, exploring the impact of the number of negative samples sampled for each positive sample on model performance. The results indicate that, as the number of negative samples increases, there is a threshold beyond which further increases in negative sample quantity actually lead to a decline in model performance. This experimental outcome suggests that focusing solely on increasing the quantity of negative samples without regard for their quality can adversely affect model training performance due to the large introduction of false negatives and easy-to-separate negatives.

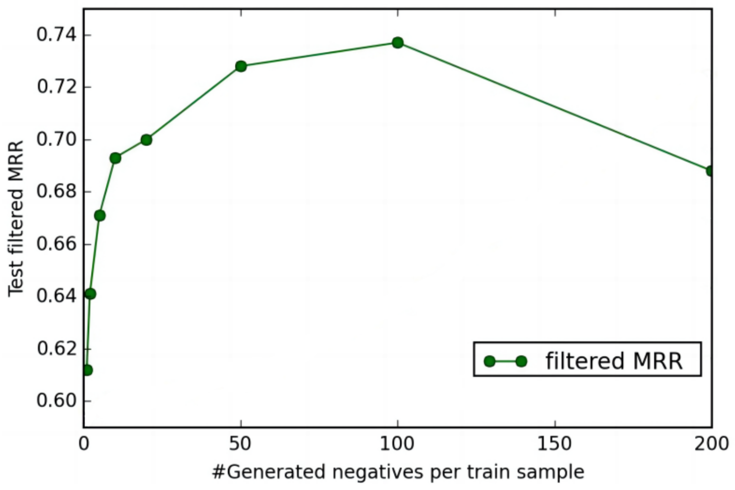

Figure 2. Experimental results of the impact of negative sampling number on model training.

Additionally, existing knowledge graphs face the problem of data imbalance, where entities and relations follow a long-tail distribution. A small number of entities and relations frequently appear in the knowledge graph, while the vast majority appear in only a limited number of triples. In current knowledge graph embedding training, there is no distinction made between the frequencies of different entities and relations, leading to a model training process that focuses more on high-frequency entities and relations, while low-frequency long-tail entities and relations receive relatively less attention. This results in insufficient training for long-tail entities and relations, which also adversely affects the improvement of model training performance. For example, reference [11] found that, in link prediction tasks using the TransE model on the benchmark dataset FB15K-237, the high-frequency entity "UK", appearing 1953 times, achieved a mean reciprocal rank (MRR) score of 0.408, while the low-frequency entity "pizza", appearing only 8 times, received an MRR score of just 0.194. This experiment demonstrates the negative impact of long-tail entities and relations on model training performance.

To address this issue, we propose a universal knowledge graph embedding framework based on high-quality negative sampling and weighting, called HNSW-KGE. First, we select the non-sampling framework NS-KGE [11] model for pre-training. This allows us to quickly learn a relatively high-quality initial embedding vector for entities and relations in the knowledge graph based on the global features of the dataset, providing support for the subsequent construction of a high-quality candidate negative sample set. Next, we design a strategy for constructing a high-quality candidate negative sample set. To overcome the common problem faced by both traditional negative sampling and non-sampling training methods of including low-quality negative samples in training, we use the embedding vectors obtained from pre-training to sample a certain number of negative samples—neither false negatives nor easy-to-separate negatives—for all positive triples. This forms a candidate negative sample set that guarantees high-quality negative samples for the subsequent model training. Finally, we adopt a loss function weighting strategy based on the frequency of triples. Weighting the loss function has been proven to be an effective method for alleviating the impact of long-tail entities on training. Therefore, we apply a weighting factor based on the frequency of triples to the loss function to adjust for the negative impacts of long-tail entities and relations on model training. To evaluate the performance of the HNSW-KGE training framework, experiments were conducted on four KGE models—TransE [4], DistMult [5], SimplE [6], and ComplEx [7]—demonstrating the effectiveness of the HNSW-KGE framework.

The main contributions of this paper can be summarized as follows.

- We propose a new knowledge graph embedding training framework, HNSW-KGE, which enhances the quality of negative sampling through pre-training based on the NS-KGE model and a high-quality negative sampling strategy. By weighting the loss function, we alleviate the adverse effects of long-tail entities and relations on model training, thereby improving model performance.
- The proposed HNSW-KGE training framework is universal for negative-sampling-based KGE models, making it applicable to various types of knowledge embedding models. We demonstrate the application of this training framework using four KGE models, i.e., TransE [4], DistMult [5], SimplE [6], and ComplEx [7]
- We conduct validation experiments using multiple knowledge embedding models on two benchmark datasets, proving the effectiveness of the proposed model framework.

2. Related Work

In this section, we will outline traditional knowledge graph embedding models, methods of negative sampling in the field of knowledge graph embeddings, and non-sampling-based knowledge graph embedding methods.

2.1. Traditional Knowledge Graph Embedding Models

Existing KGE models can be roughly categorized into three types.

Geometric Transformation Methods. These methods model the semantic relations between entities and relations through geometric transformations. The most classic model, TransE [4], views relations as a translation transformation between the head and tail entity vectors in the same space. Models like TransH [12], TransR [13], TransD [14], and RotatE [15] build upon TransE to handle more complex relations.

Tensor Decomposition Methods. These models construct scoring functions through tensor decomposition to measure the probability of triple validity. A widely used model, DistMult [5], models the scoring function as the inner product of the head and tail entity vectors with a diagonal matrix of relations. Subsequent models such as ComplEx [7], HolE [16], ANALOGY [17], and SimplE [6] extend DistMult to achieve better embedding performance.

Neural Network Methods. This category utilizes deep neural networks like CNNs, RNNs, and GNNs to model the complex relations between entities and relations through nonlinear transformations. Models such as ConvE [18] and ConvKB [19] use CNNs to capture latent semantics between entities and relations; Gardner et al. [20] used RNNs to model relation paths to capture longer relation dependencies in KGs; R-GCN [21], RGHAT [22], and CompGCN [23] leverage GNNs to capture structured information within KGs.

2.2. Negative Sampling Methods in Knowledge Graph Embedding

Negative sampling methods in knowledge embedding models can be categorized into three types: random negative sampling, adversarial sampling, and additional data-enhanced sampling. Each will be introduced below.

Random Negative Sampling. This is the most basic and widely used negative sampling method. For instance, TransE [4] randomly selects an entity from the entity set to replace in a positive triple to generate a negative sample. To improve training effectiveness, TransH [12] introduces a Bernoulli sampling strategy, sampling negative triples with probabilities proportional to the relation types in the knowledge graph. However, random negative sampling often results in overly simple negative samples (easy-to-separate negatives), which do not provide useful information for model optimization, leading to local optima.

Adversarial Negative Sampling. These models are usually based on the GAN framework, where the generator module samples negative triplets and the discriminator module distinguishes between positive and negative triplets to sample better hard-to-distinguish negative samples. KBGAN [9] uses policy-gradient-based reinforcement learning for discrete negative sampling to confuse the discriminator. IGAN [8] uses a two-layer fully connected neural network as a generator. GraphGAN [24] proposes a new generator, graph softmax, that uses graph structure information and neighborhood information to overcome the limitations of the traditional softmax activation function. KSGAN [25] uses an additional neighborhood aggregator component, namely the neighborhood knowledge selective adversarial network (NKSGAN), to generate high-quality negative samples. KCGANs [26] use a game theory framework to enhance the understanding of the underlying knowledge graph structure. The RUGA [27] method uses labeled triplets, unlabeled triplets, and soft rules to promote iterative learning of sample information. GN+DN [28] aims to generate previously unseen but credible samples. However, such models usually have shortcomings, such as high complexity and unstable training, and cannot solve the problem of false negative samples well.

These models [8,9,22,29], often based on GAN principles, utilize the semantic information of samples in the embedding space to select negative samples, aiming to sample more difficult-to-separate negatives. However, they suffer from complex frameworks, unstable training results, and longer training times. To address this, RotatE [15] proposes a self-adversarial sampling strategy, dynamically adjusting the sampling probabilities of negative samples during the training process to focus on more challenging negatives.

Additional Data-Enhanced Sampling. This approach incorporates external information during negative sampling to increase the sampling probability of difficult-to-separate negatives. Ref. [30] integrates prior knowledge in the form of type constraints to assist in generating high-quality negative samples. Ref. [31] uses other new conditional constraints based on OWL limitations. Ref. [32] incorporates ontology reasoning into the negative sampling process to enhance embedding accuracy. CAKE [33] selectively filters effective and high-quality negative samples through the use of common-sense knowledge.

Overall, regardless of the type of negative sampling method used, the variability of negative samples adopted during different sampling rounds can affect the stability of model training. Additionally, the inevitable introduction of false negatives can further impact the accuracy of model training. The HNSW-KGE negative sampling training framework that we proposed continuously updates the knowledge graph embedding during pre-training and model training, builds a strategy based on a high-quality candidate negative sample set, and uses the effective information of the model's own training process to sample high-quality negative samples without adding additional sample information. It achieves a training effect similar to self-adversarial negative sampling and can also effectively alleviate the adverse effects of false negative samples.

2.3. *Non-Sampling-Based Knowledge Graph Embedding Models*

To overcome the high training complexity and prediction bias associated with excessive reliance on negative sample selection, Ref. [34] proposes a training method that adds a regularization term to the loss function to prevent the model from assigning high scores to all triples, thereby eliminating the need for negative sampling. Although this approach does not explicitly use negative sampling, the calculation of the regularization term implicitly utilizes all possible triples in the knowledge graph, including both positive and negative triples. KG-NSF [35] proposes a method that does not require negative samples for training, directly learning knowledge graph embeddings based on the cross-correlation matrix of the embedding vectors of entities and relations. However, this method has high space complexity due to the need for matrix multiplication. The non-sampling knowledge graph embedding framework, NS-KGE [36], uses all non-positive triples (composed of entities and relations) as negative triples for training, thus addressing the shortcomings of negative sampling methods by skipping the sampling process. However, this method introduces false negative samples and a large number of easy-to-separate negatives, which can affect the accuracy and convergence speed of model training.

3. Problem Description

In this section, we will formally describe the research problem under investigation. Table 1 lists the mathematics symbols used.

Table 1. Mathematics symbols in this paper.

Symbol	**Description**
G	Knowledge graph
E	Set of entities in the knowledge graph
R	Set of relations in the knowledge graph
S	Set of positive triples
S'	Set of negative triples
(h, r, t)	Positive triple
(h', r, t')	Corresponding negative triple to (h, r, t)
h, t	Head entity and tail entity of the triple in the knowledge graph
r	Relation of the triple in the knowledge graph
$\mathbf{h}, \mathbf{t}$	Embedding vectors of the head entity and tail entity in the triple
$\mathbf{r}$	Embedding vector of the relation in the triple
d	Dimension of the embedding vectors
W	Weight of the loss function
$f(h, r, t)$	Scoring function for the triple
$\mathcal{L}$	Loss function of the model

Give a knowledge graph $G = (E, R, S)$, where a triple $(h, r, t) \in S, h, t \in E, r \in R$.

First, we perform negative sampling. To construct a candidate negative sample set required for model training, a common approach is to randomly replace the head entity or tail entity of the triple (h, r, t) with entities from the entity set E to obtain corresponding negative triples (h', r, t'). Thus, the candidate negative sample set S' can be formally described as $S' \in \{(h', r, t) \mid (h, r, t) \in S \wedge (h', r, t) \notin S \wedge h \neq h' \wedge h' \in E\} \cup \{(h, r, t') \mid (h, r, t) \in S \wedge (h, r, t') \notin S \wedge t \neq t' \wedge t' \in E\}$. Based on a specific strategy, such as random negative sampling or hard negative sampling, a portion of negative triples are sampled from the candidate negative sample set for training.

Next, we compute the loss. According to the model's scoring function, we calculate the scoring function values for the positive triple (h, r, t) and its corresponding negative triple (h', r, t'), denoted as $f(h, r, t)$ and $f(h', r, t')$, respectively. The loss for the triples is then computed based on these scoring function values, resulting in the model's loss function $\mathcal{L}$.

The model's loss function can generally be divided into two categories: loss functions based on margin hyperparameters and loss functions based on cross-entropy. The loss function based on margin hyperparameters is defined as

$$\mathcal{L} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} [\gamma + f(h, r, t) - f(h', r, t')]_+, \tag{1}$$

The parameter γ is the margin hyperparameter. Translation-based models, such as TransE and TransH, primarily utilize loss functions based on margin hyperparameters.

The loss function based on cross-entropy is calculated as

$$\begin{aligned}\mathcal{L} = \sum_{(h,r,t)\in S} &(\log(1 + \exp(-f(h, r, t))))+ \\ &\sum_{(h'_i,r,t'_i)\in S'} (\log(1 + \exp(f(h'_i, r, t'_i)))),\end{aligned} \tag{2}$$

Models based on semantic matching, such as DistMult, SimplE, and ComplEx, predominantly utilize loss functions based on cross-entropy.

Finally, we perform optimization training. Various optimization algorithms, such as stochastic gradient descent (SGD) and adaptive gradient algorithm (AdaGrad), are used to optimize the loss function L. The goal is to obtain the optimal parameters for the model, including the embedding vectors $\mathbf{h}, \mathbf{r}, \mathbf{t}$ for the entities h, t and the relation r.

4. Proposed Method

In this section, we will provide a detailed introduction to the HNSW-KGE training framework. We will first introduce the NS-KGE model used for pre-training, followed by the strategy for constructing high-quality candidate negative sample sets. Next, we will discuss the weighting strategy for the loss function based on triple frequencies. Finally, we will explore the application of this training framework across various knowledge graph embedding models.

4.1. Pre-Training Based on NS-KGE

To provide a basis for constructing the high-quality candidate negative sample set, we select the non-sampling knowledge graph embedding framework NS-KGE as the pre-training model. This choice allows us to quickly obtain an initial representation of knowledge graph embeddings. The non-sampling knowledge graph embedding framework NS-KGE can be applied to knowledge embedding models based on square loss functions or models that can be converted to square loss. Compared to traditional knowledge graph embedding models, NS-KGE offers relatively higher training efficiency and some improvement in training accuracy. The fundamental idea behind NS-KGE is based on the closed-world assumption, where all triples composed of entities and relations present in the dataset but not included in the dataset are treated as negative samples during model training. The loss function for optimization is defined as the least squares loss between the true label values of the positive and negative triples and the predicted values based on the scoring function.

To address the issue of high model training complexity due to the large dataset size, the NS-KGE model mathematically derives a way to reduce the complexity of the loss function. The loss function defined for optimization training in the NS-KGE framework can be expressed as

$$\mathcal{L} = \sum_{(h,r,t)\in S} C_{hrt}[f_r(h,t) - \hat{f}_r(h,t)]^2, \tag{3}$$

In the NS-KGE model, C_{hrt} represents the weight associated with the corresponding triple (h,r,t). The function $f_r(h,t)$ denotes the true label value for the triple (h,r,t). Specifically, this is defined as follows. For positive triples (h,r,t), we define $f_r(h,t) = 1$. For negative triples (h,r,t), we define $f_r(h,t) = 0$. $\hat{f}_r(h,t)$ represents the predicted label value based on the scoring function.

NS-DistMult is a sub-model derived from applying the NS-KGE framework to the DistMult model. The prediction function for the DistMult model is defined as follows:

$$\hat{f}_r(h,t) = \sum_i^d \mathbf{h}_i \cdot \mathbf{r}_i \cdot \mathbf{t}_i. \tag{4}$$

To reduce the time and space complexity of the model, we combine Equation (4) (the prediction function) with Equation (3) (the loss function) for mathematical derivation. This results in the least squares loss function for NS-DistMult being defined as follows:

$$\mathcal{L} = \mathcal{L}^P + C^- \sum_i^d \sum_j^d (\sum_{h\in E} \mathbf{h}_i \cdot \mathbf{h}_j)(\sum_{r\in R} \mathbf{r}_i \cdot \mathbf{r}_j)(\sum_{t\in E} \mathbf{t}_i \cdot \mathbf{t}_j), \tag{5}$$

wherein

$$\mathcal{L}^P = \sum_{r\in R} \sum_{h\in E} \sum_{t\in E^+_{hr}} (C^+(\hat{f}_r(h,t)^2 - 2f_r(h,t)\hat{f}_r(h,t)) - C^-\hat{f}_r(h,t)^2). \tag{6}$$

Similarly, NS-TransE is a sub-model derived from applying the NS-KGE framework to the TransE model. According to the scoring function of TransE,

$$f(h,r,t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|, \tag{7}$$

By making a transformation, the prediction function of the TransE model can be expressed as

$$\hat{f}_r(h,t) = 1 - \frac{1}{3} f(h,r,t) = 1 - \frac{1}{3}\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|. \tag{8}$$

After mathematical derivation and simplification, the least squares loss function for NS-TransE can be expressed as

$$\mathcal{L} = \mathcal{L}^P + \frac{4C^-}{9} \sum_i^d \sum_j^d \begin{pmatrix} |E| \sum_{\mathbf{r}\in R} \mathbf{r}_i \mathbf{r}_j \sum_{\mathbf{t}\in E} \mathbf{t}_i \mathbf{t}_j + |E| \sum_{\mathbf{h}\in E} \mathbf{h}_i \mathbf{h}_j \sum_{\mathbf{r}\in R} \mathbf{r}_i \mathbf{r}_j + \\ |R| \sum_{\mathbf{h}\in E} \mathbf{h}_i \mathbf{h}_j \sum_{\mathbf{t}\in E} \mathbf{t}_i \mathbf{t}_j - 2 \sum_{\mathbf{r}\in R} \mathbf{r}_i \mathbf{r}_j \sum_{\mathbf{h}\in E} \mathbf{h}_i \sum_{\mathbf{t}\in E} \mathbf{t}_j \end{pmatrix}, \tag{9}$$

where the calculation of $\mathcal{L}^P$ is shown in Equation (6).

In terms of training performance, NS-KGE generally achieves better results compared to the original baseline models. For example, on the FB15K237 dataset, the Hit@10 metric improves from 0.316 for TransE to 0.447 for the NS-TransE model. This improvement is due to the fact that the original baseline models use negative sampling methods that randomly sample only a portion of the negative sample information from the dataset, potentially overlooking some important negative samples. In contrast, the NS-KGE framework involves all negative samples from the dataset in the training process.

Regarding training efficiency, NS-KGE also shows a significant improvement in training speed compared to the original baseline models. For instance, when compared to TransE on the FB15K237 dataset, the training speed of NS-TransE is approximately ten times faster. This is because the original negative-sampling-based models spend most of their time on negative sampling, whereas the NS-KGE framework skips the negative sampling process altogether.

However, NS-KGE has certain limitations. First, its scalability is limited: the model uses mathematical derivation to reduce the complexity of the loss function, but, due to the complexity of the derivation process, it is applicable only to a limited number of models, restricting its range of application. Second, it is prone to introducing false negative samples. The model treats all unobserved triples in the dataset as negative samples during training, leading to many factual triples being considered negative samples due to data sparsity, which affects the accuracy of model training. Third, the non-sampling approach can introduce a large number of easy-to-separate negative samples during training, impacting the convergence speed and accuracy of the model.

For these reasons, we use NS-KGE solely for pre-training as a starting point for training all models, allowing for the quick acquisition of relatively high-quality initial embedding vectors, which will provide a reference for constructing high-quality candidate negative sample sets. The high-quality candidate negative sets will remove these false negatives to a large extent. Additionally, considering both training effectiveness and efficiency, we select the NS-TransE sub-model from NS-KGE as the pre-training model.

4.2. High-Quality Candidate Negative Sample Set Construction

To provide high-quality negative samples for training based on negative sampling, we first obtain an initial set of high-quality embedding vectors for entities and relations in the dataset. Next, we will construct a candidate negative sample set for all positive triples according to our designed strategy, which will serve as a candidate pool for negative samples, allowing us to sample high-quality negatives for model training, thereby enhancing the performance of the training process. The following sections will detail the strategy for constructing high-quality candidate negative sample sets.

The candidate negative sample set includes a candidate head entity set, derived from replacing the head entity, and a candidate tail entity set, derived from replacing the tail entity. For each triple, we sample a total size of $N = N_h + N_t$ for the candidate negative sample set, where N_h is the size of the candidate head entity set and N_t is the size of the candidate tail entity set.

When setting N_h and N_t, we consider the type of relation corresponding to the triple, such as one-to-many or many-to-one relations. Specifically, for triples with a one-to-many relation type, the uniqueness of the head entity means that negative samples constructed by replacing the head entity are unlikely to be false negatives. Therefore, we can afford to include more candidate head entities, resulting in a larger size for the candidate head entity set compared to the candidate tail entity set. Conversely, for triples with a many-to-one relation type, we can consider including more candidate tail entities, leading to a larger size for the candidate tail entity set compared to the candidate head entity set. We define the size of the candidate head entity set as N_h and the size of the candidate tail entity set as N_t, respectively.

$$N_h = N\frac{t_{rh}}{h_{rt} + t_{rh}}, \tag{10}$$

$$N_t = N\frac{h_{rt}}{h_{rt} + t_{rh}}, \tag{11}$$

where t_{rh} refers to the average number of tail entities connected to the head entities of all triples with the relation r in the knowledge graph, and h_{rt} refers to the average number of head entities connected to the tail entities of all triples with the relation r.

Next, using TransE as an example, we will detail the process of constructing the candidate head entity set for the triple (h, r, t).

Step 0: Initialization: construct a data structure of the form $\{(r,t), \{(h_l, f(h'_l, r, t)), \ldots (h'_{N_h}, f(h_{N_h}, r, t))\}\}$, where the array has a size of N_h. Initialize $h_1, \ldots, h_{N_h}$ to be empty. Here, $f(h', r, t)$ represents the corresponding score function value, and the scoring function for the TransE model is defined as shown in Equation (7). Initialize the score values to a relatively large number, such as 100.

Step 1: Construct candidate negative triples: randomly select an entity h' from the entity set to replace h in (h, r, t), resulting in the negative triple (h', r, t).

Step 2: Check if (h', r, t) exists in the dataset. Determine if (h', r, t) is presented in the dataset. If it is, return to Step 1. If not, proceed to Step 3. This step is necessary to exclude any candidate negative triples that are known positive samples.

Step 3: Check if (h', r, t) is a false negative sample:

$$f(h', r, t) < f(h, r, t) + u, \tag{12}$$

Determine whether the condition holds; if it does, this indicates that (h', r, t) may be a false negative sample, and return to Step 1. Otherwise, proceed to Step 4. Here, $u \geq 0$ is a hyperparameter. This step aims to exclude the candidate negative sample if it is identified as a false negative.

Step 4: Filter candidate head entities. Iterate to check if the score function value $f(h', r, t)$ is smaller than the score function values of all negative triples constructed from the candidate head entities. If $f(h', r, t) < f(h'_i, r, t)$ for any negative triple (h'_i, r, t) in the candidate head entity set, it indicates that (h', r, t) is a more challenging negative triple than (h'_i, r, t). In this case, replace $(h'i, f(h'_i, r, t))$ with $(h', f(h', r, t))$. Otherwise, return to Step 1. This step aims to select the N_h entities that are the most difficult to distinguish, ranking them by their score function values from smallest to largest, and using these N_h entities as the candidate head entities.

Step 5: Output. Once all entities in the entity set have been iterated through, output the candidate head entity set of size N_h for the triple (h, r, t).

The construction method for the candidate tail entity set for the triple (h, r, t) is similar. Ultimately, we obtain a candidate negative sample set of size $N = N_h + N_t$ for the triple (h, r, t). This process constructs the corresponding candidate negative sample set for all triples in the dataset, resulting in a high-quality candidate negative sample set for that dataset.

Additionally, to mitigate the potential quality issues of the candidate negative sample set due to the performance limitations of the pre-trained model NS-TransE, we will dynamically update the negative sample set multiple times during training. This updating process is based on the current knowledge graph embedding vectors and follows the aforementioned steps. This approach ensures that we provide high-quality negative samples for the subsequent optimization training based on the loss function.

The whole process of dynamically updating the candidate negative sample set is similar to the process of self-adversarial negative sampling, that is, selecting negative samples that can fit the data distribution of the current training process, while the hyperparameters of the model are tuned on the validation set rather than on the training sets. By doing so, it could avoid over-fitting issues.

4.3. Weighting the Loss Function Based on Triple Frequencies

The loss function L for model training is defined as follows:

$$L = \sum_{(h,r,t) \in S} L(h, r, t) + \sum_{(h',r,t')' \in S'} L(h', r, t'), \tag{13}$$

The tuples (h, r, t) and (h', r, t') represent positive and negative triplets, respectively, while S and S' denote the sets of positive and negative triplets.

To mitigate the impact of long-tail entities and relations on model training, it is essential to increase the focus on these long-tail entities and relations, thereby improving the training performance of knowledge graph embedding models. Therefore, we introduce a weight factor based on triplet frequency into the loss function. This allows us to assign different weights to the loss function of triplets according to the frequency of entities and relations in the dataset. Specifically, we assign lower weights to entities and relations that occur more frequently while giving higher weights to those that are less frequent. This approach enhances the training focus on long-tail entities and relations, with the goal of learning better embeddings. We define the weight factor W_f based on triplet frequency as follows:

$$W_f = \frac{1}{3}(w_h + w_r + w_t), \tag{14}$$

$$w_h = \frac{f_e^0}{f_h + 1}, w_r = \frac{f_r^0}{f_r + 1} w_t = \frac{f_e^0}{f_t + 1}, \tag{15}$$

In this context, f_h, f_t and f_r represent the frequencies of entities h, t and relation r in the dataset, which, respectively, represent the median frequency for entities and relations in the dataset, which is defined according to the Pareto principle (80/20 rule), where we consider the frequency of the top 20% of entities and relations ranked by frequency.

In the calculation formula for w_h in Equation (15), the denominator includes $f_h + 1$ to avoid cases where certain entities have a frequency of zero in the dataset when calculating the frequency weight for entities in negative triplets. The situations for w_t, w_r are similar.

The weight factor W_f based on triplet frequency can be understood as follows: for low-frequency triplets—where both entities and the relation in the triplet have frequencies below the median—the weight assigned to these low-frequency entities or relations will be greater than 1, according to Equation (15). Consequently, the W_f value in the loss function for low-frequency triplets will also be greater than 1, thereby increasing the model's focus on these low-frequency entities and relations during training.

Conversely, for high-frequency triplets—where all entities and relations have frequencies above the median—the weights assigned will be less than 1, following the same logic in Equation (15). Thus, the W_f in the loss function for high-frequency triplets will also be less than 1, reducing the model's emphasis on these high-frequency entities and relations during training.

To illustrate this with an analogy related to student learning, consider the knowledge points that frequently appear in practice exercises, akin to high-frequency triplets. Since these topics are practiced often, focusing on them may not significantly enhance performance, allowing students to allocate less time to them. In contrast, knowledge points that rarely appear in exercises resemble low-frequency triplets. Because these topics are less frequently practiced, they require more attention, and focusing on them is key to improving overall performance.

The weight factor W_f based on the frequency of triplets is applied to Formula (13), resulting in

$$L_f = \sum_{(h,r,t)\in S} W_f \cdot L(h,r,t) + \sum_{(h',r,t')'\in S'} W_f \cdot L(h',r,t'), \tag{16}$$

where the calculation of W_f is shown in Formula (14).

4.4. Applications of HNSW-KGE on Different Models

In this section, we will illustrate the application of our proposed HNSW-KGE training framework on four typical knowledge graph embedding models: TransE, DistMult, SimplE, and ComplEx. Although we choose HNSW-KGE for pre-training, it will not affect the choice of other knowledge graph embedding models in the following training steps. We choose NS-KGE only for pre-training, which constructs a candidate negative sample set. It will not affect the following phases of the whole framework. During high-quality negative sample set construction, other knowledge graph embedding models are applied to calculate the scoring function. There is no limit of the range of knowledge graph embedding models in this phase.

For TransE, the scoring function for a triplet $f(h,r,t)$ is defined as shown in Formula (7). The TransE model is a translation-based model, and such models typically use a margin-based loss function as shown in Formula (1). To facilitate the weighting of the loss function for positive and negative triplets, we modify the margin-based loss function to

$$\mathcal{L} = -\Big(\sum_{(h,r,t)\in S} log\sigma(\gamma - f(h,r,t)) + \sum_{(h'_i,r,t'_i)\in S'} log\sigma(f(h'_i,r,t'_i) - \gamma)\Big), \tag{17}$$

where γ is the margin parameter, σ is the sigmoid activation function, and (h'_i, r, t'_i) is the corresponding negative triplet of (h,r,t) used for training. Combining this with Formula (16) to weight the loss function, we obtain the loss function $\mathcal{L}$ for this type of model as

$$\mathcal{L} = -\sum_{(h,r,t)\in S} W_f \cdot log\sigma(\gamma - f(h,r,t)) - \sum_{(h'_i,r,t'_i)\in S'} W_f \cdot log\sigma(f(h'_i,r,t'_i) - \gamma). \tag{18}$$

For DistMult, the scoring function for a triplet $f(h, r, t)$ is defined as

$$f(h,r,t) = \boldsymbol{h}^T \cdot \mathrm{diag}(\boldsymbol{r}) \cdot \boldsymbol{t}. \tag{19}$$

For SimplE, the scoring function for a triplet $f(h, r, t)$ is defined as

$$f(h,r,t) = -\frac{1}{2}(\boldsymbol{h}^{\mathrm{T}}(\boldsymbol{r}\odot\boldsymbol{t}) + \boldsymbol{t}^{\mathrm{T}}(\boldsymbol{r}^{-1}\odot\boldsymbol{h})). \tag{20}$$

For ComplEx, the scoring function for a triplet $f(h, r, t)$ is defined as

$$f(h,r,t) = -\begin{pmatrix} \boldsymbol{h}_{\mathrm{re}}^{\mathrm{T}}(\boldsymbol{r}_{\mathrm{re}}\odot\boldsymbol{t}_{\mathrm{re}}) + \boldsymbol{h}_{\mathrm{im}}^{\mathrm{T}}(\boldsymbol{r}_{\mathrm{re}}\odot\boldsymbol{t}_{\mathrm{im}}) \\ +\boldsymbol{h}_{\mathrm{re}}^{\mathrm{T}}(\boldsymbol{r}_{\mathrm{im}}\odot\boldsymbol{t}_{\mathrm{im}}) - \boldsymbol{h}_{\mathrm{im}}^{\mathrm{T}}(\boldsymbol{r}_{\mathrm{im}}\odot\boldsymbol{t}_{\mathrm{re}}) \end{pmatrix}. \tag{21}$$

Models such as DistMult, SimplE, and ComplEx are based on semantic matching. The loss function for these models is defined based on cross-entropy, as shown in Formula (2).

Combining this with Formula (16) to weight the loss function, we obtain the loss function $\mathcal{L}$ for these models as

$$\mathcal{L} = \sum_{(h,r,t)\in S} W_f \cdot (log(1+\exp(-f(h,r,t)))) + \sum_{(h'_i,r,t'_i)\in S'} W_f \cdot log\left(1+\exp\left(f\left(h'_i,r,t'_i\right)\right)\right). \quad (22)$$

The overall process of our proposed HNSW-KGE framework is illustrated in Figure 3. First, based on the pre-trained embedding vectors and the designed high-quality candidate negative sample set strategy, we sample a high-quality candidate negative sample set for all positive triplets. Next, depending on the different models, we compute the scoring function values for the triplets and their corresponding negative triplets. After that, we gather relevant frequency data from the dataset and apply Formulas (14) and (15) to obtain the frequency-based weight Wf for each triplet.

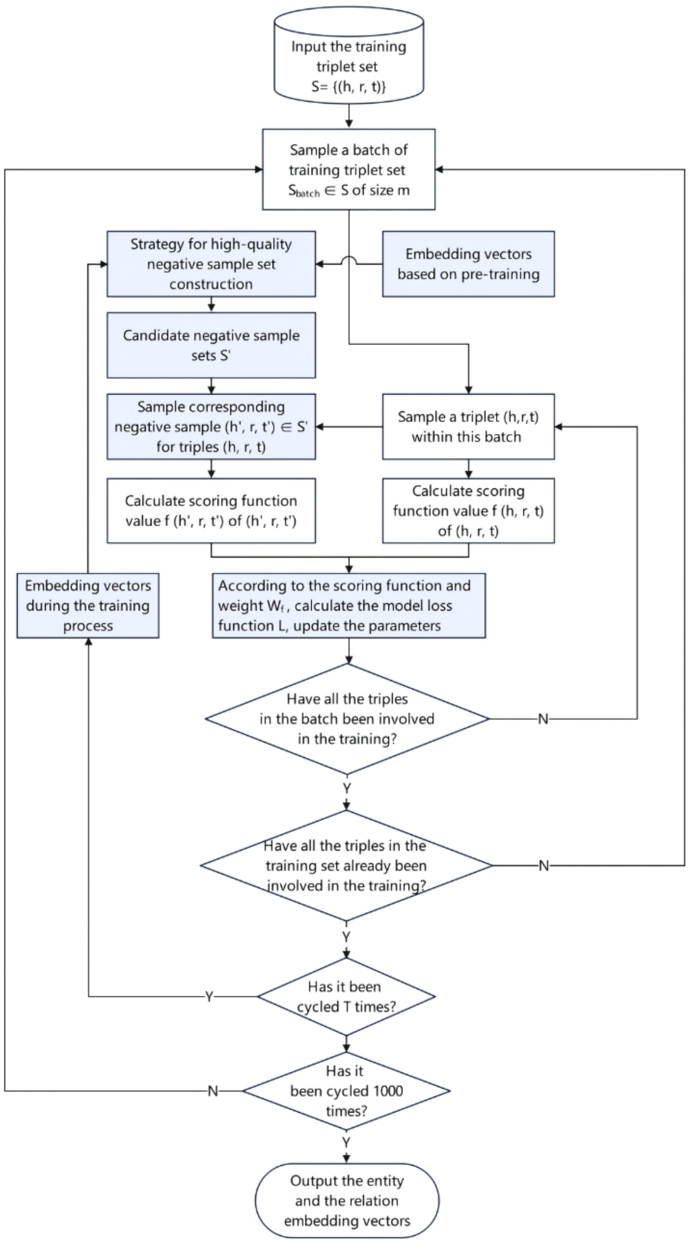
Input the training triplet set S= {(h, r, t)}
Sample a batch of training triplet set $S_{batch} \in$ S of size m
Strategy for high-quality negative sample set construction
Embedding vectors based on pre-training
Candidate negative sample sets S'
Sample corresponding negative sample (h', r, t') ∈ S' for triples (h, r, t)
Sample a triplet (h,r,t) within this batch
Calculate scoring function value f (h', r, t') of (h', r, t')
Calculate scoring function value f (h, r, t) of (h, r, t)
Embedding vectors during the training process
According to the scoring function and weight W_f, calculate the model loss function L, update the parameters
Have all the triples in the batch been involved in the training?
N
Y
Have all the triples in the training set already been involved in the training?
N
Y
Has it been cycled T times?
Y
Has it been cycled 1000 times?
N
Y
Output the entity and the relation embedding vectors

Figure 3. Overall flowchart of the HNSW-KGE framework.

Then, using the scoring function values and W_f, we calculate the loss L for the triplet set. For translation-based models, the loss is computed according to Formula (18), while, for semantic matching models, it is computed according to Formula (22), yielding the final loss function for the training framework. Finally, we optimize the model's loss function using stochastic gradient descent (SGD) to learn the optimal parameters of the model, including the optimal embedding vectors $\mathbf{h}, \mathbf{r}, \mathbf{t}$.

To avoid over-fitting, we enforce a constraint during model training that the 2-norm of all entities and relations in the dataset is less than or equal to 1, i.e., $||\mathbf{h}||_2 \leq 1$, $||\mathbf{r}||_2 \leq 1$, $||\mathbf{t}||_2 \leq 1$.

5. Experimental Setup

We evaluate the performance of the HNSW-KGE framework through a link prediction task. In this section, we first introduce the datasets used in the experiments, followed by the experimental design. Next, we describe the evaluation metrics, then present the comparison models, and finally discuss the parameter settings.

5.1. Datasets

We will conduct experiments on two benchmark datasets, FB15K237 and WN18RR, used in the NS-KGE framework [11] to evaluate the training performance of the HNSW-KGE framework. Both datasets are commonly used benchmarks in knowledge graph embedding. FB15K237 was created to prevent information leakage during testing by removing reversible relation triplets from the Freebase subset FB15K. Similarly, WN18RR was designed to avoid information leakage by deleting reversible relation triplets from the WordNet subset WN18. The basic statistical information for these two datasets is provided in Table 2.

Table 2. Dataset statistics.

Datasets	**FB15K237**	**WN18RR**
#Entities	14,541	40,943
#Relations	237	11
#Triples	292,581	89,969

5.2. Experimental Design

In the link prediction task, the general setup method for the dataset involves dividing the set of triplets into a training set, a validation set, and a test set, approximately accounting for 85%, 5%, and 10%, respectively. To avoid over-fitting, the hyperparameters of the model are tuned on the validation set rather than the training set. The specific experimental steps for link prediction are as follows.

First, the optimal parameters of the model are obtained through training on the training set and validation set, including the optimal embedding vectors for all entities and relations. Then, the link prediction task is performed on the test set to evaluate the model's ability to predict the missing head or tail entity in the triples. Specifically, given (h, r), the model predicts t, or, given (r, t), it predicts h. To illustrate the evaluation of the model's ability to predict the tail entity, for a test triple (h, r, t), each entity x in the entity set is used to replace the tail entity t to create a set of candidate triples (h, r, x). The scores of these candidate triples are calculated based on the scoring function $f(h, r, x)$ defined by the model. For example, the TransE model uses Formula (7), while the DistMult model uses Formula (19), etc. The scores are sorted in ascending order, allowing us to determine the rank of the test triple (h, r, t) among all candidate triples, denoted as rank(t). Similarly, to evaluate the model's ability to predict the head entity, we rank the entity test triple (h, r, t) among all candidate triples (x, r, t), denoted as rank(h).

Finally, based on the values of rank(h) and rank(t) for the test triple (h, r, t), the evaluation metrics are calculated according to the defined formulas. Additionally, a "Filter" setting is applied. Specifically, before evaluating the ranking of each test triple, the triples existing in the training, validation, and test sets are removed from the set of candidate triples. This is because those candidate triples are essentially positive triples, and it is reasonable for their scores based on the scoring function to rank higher than the original test triple. Therefore, it is necessary to exclude these candidate triples before obtaining the score rankings for each test triple to eliminate the influence of "interference" factors.

5.3. Evaluation Metrics

In the link prediction task, several evaluation metrics are used to assess the quality of the knowledge graph embeddings learned by the model and to evaluate the model's training performance. MR: this is the average rank of all test triples. It is defined as follows:

$$\mathrm{MR} = \frac{1}{2|S_t|} \sum_{h,r,t \in S_t} (\mathrm{rank}(h) + \mathrm{rank}(t)), \tag{23}$$

In this context, S_t represents the test tuple set and $|S_t|$ denotes the size of the test tuple set.

MRR is defined as the average of the reciprocal ranks of all test tuples. Specifically, it can be defined as follows:

$$\mathrm{MRR} = \frac{1}{2|S_t|} \sum_{h,r,t \in S_t} (\frac{1}{\mathrm{rank}(h)} + \frac{1}{\mathrm{rank}(t)}). \tag{24}$$

The Hits@K metric is defined as the ratio of the number of test tuples ranked at or below K to the total number of test tuples. In this experiment, we utilize metrics such as Hits@1, Hits@3, and Hits@10. Higher values of MRR and Hits@N, along with lower values of mean rank (MR), indicate a better training performance of the model; conversely, lower MRR and Hits@N values, along with higher MR values, suggest a poorer model performance.

5.4. Baseline Model

We categorize the comparison models into three types.

(1) Original Baseline Models: These are knowledge graph embedding models based on traditional negative sampling. We selected four classic models—TransE, DistMult, SimplE, and ComplEx—as comparison models. The experimental results for link prediction from these models are obtained directly from published papers.

TransE [4]: A translation-based model that models relation vectors as "translation" operations between head and tail entity vectors in the embedding space. It measures the authenticity of triples by assessing the distance between the translated head and tail entity vectors.

DistMult [5]: A semantic matching model that utilizes matrix factorization for knowledge graph embedding learning. It models all triples in the knowledge graph as a three-dimensional tensor and derives entity and relation embeddings through tensor decomposition.

SimplE [6]: A semantic matching model that builds on DistMult by incorporating inverse relations, serving as an interpretable bilinear model.

ComplEx [7]: A semantic matching model that captures the interaction information between head and tail entities and relations using Hermitian dot products in complex space.

(2) Models Using the NS-KGE Framework: These are versions of the original baseline models trained using the NS-KGE framework, including NS-TransE, NS-DistMult, NS-SimplE, and NS-ComplEx. The link prediction experimental results for these models are also sourced from published papers.

(3) Models Using the HNSW-KGE Framework: These are versions of the original baseline models trained using the HNSW-KGE framework, including HNSW-TransE, HNSW-DistMult, HNSW-SimplE, and HNSW-ComplEx.

5.5. Parameter Settings

The learning rate for stochastic gradient descent (SGD) is set to $\lambda \in \{0.0001, 0.001, 0.01, 0.1\}$, the embedding dimension $d \in \{100, 300, 500, 1000\}$, the hyperparameter $u \in \{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$, and the hyperparameter $N \in \{10, 30, 50, 80, 100\}$, $\gamma \in \{0.1, 0.2, 0.5, 1,$

$2, 4, 8, 10\}$. The batch size is set to 128. The model is trained for 1000 iterations on each dataset. The optimal parameters are determined based on Hits@10 on the validation set, and the optimal parameter configurations for the two datasets are shown in Table 3.

Table 3. Optimal hyperparameter settings.

Datasets	λ	d	u	γ	N
FB15K237	0.001	100	0.05	4	50
WN18RR	0.001	100	0.05	4	50

6. Experimental Results and Analysis

In this section, to validate the effectiveness of the model, we first present the comparative experimental results of HNSW-KGE against baseline models on two public datasets, followed by an analysis. Subsequently, we conduct a parameter analysis, focusing on the impact of the number of candidate negative samples NN constructed for each positive sample in the high-quality negative sample set on experimental outcomes. We then introduce ablation experiments to analyze the influence of each module on the experimental results. Finally, we present a case study.

6.1. Comparative Experimental Results

The results of the link prediction task comparing the HNSW-KGE model with two types of baseline models are shown in Table 4, with a more visual comparison presented in Figure 4. In Table 4, NS-X refers to the version of model X that applies the NS-KGE framework while HNSW-X refers to the version of model X that applies the HNSW-KGE framework. An upward arrow (↑) indicates that a higher value is better for the evaluation metric whereas a downward arrow (↓) indicates that a lower value is better.

In the table, bold values represent the best experimental results among the baseline models while underlined values indicate the second-best results. In the figure, blue rectangles represent the average values of evaluation metrics for the original baseline models, red rectangles represent the average values for models using the NS-KGE framework, and yellow rectangles represent the average values for models using the HNSW-KGE framework.

Table 4. Link prediction experimental results.

Datasets	**FB15K237**					**WN18RR**				
Evaluation Metrics	**MRR ↑**	**MR ↓**	**Hits@10 ↑**	**Hits@3 ↑**	**Hits@10 ↑**	**MRR ↑**	**MR ↓**	**Hits@10 ↑**	**Hits@3 ↑**	**Hits@1 ↑**
DistMult	0.177	430.15	0.345	0.198	0.010	0.320	**4019.98**	0.461	0.371	0.24
NS-DistMult	0.227	361.61	0.373	0.248	0.155	**0.411**	7456.29	0.462	0.424	**0.384**
HNSW-DistMult	**0.282**	**267.70**	**0.449**	**0.306**	**0.201**	0.409	5141.6	**0.465**	**0.425**	0.377
SimplE	0.183	387.9	0.355	0.207	0.099	0.329	**3950.82**	**0.463**	0.378	0.253
NS-SimplE	0.222	364.31	0.370	0.246	0.159	0.406	7418.31	0.459	0.415	0.377
HNSW-SimplE	**0.272**	**314.1**	**0.427**	**0.295**	**0.193**	**0.416**	7504.3	0.451	**0.427**	**0.400**
ComplEx	0.240	529.42	0.415	0.271	0.151	0.39	**4673.35**	0.474	0.422	0.339
NS-ComplEx	0.243	**326.57**	0.390	0.256	0.163	0.429	7649.34	**0.485**	0.449	0.396
HNSW-ComplEx	**0.261**	365.4	**0.419**	**0.283**	**0.184**	**0.431**	6626.6	0.475	0.444	0.407
TransE	0.178	337.86	0.316	0.192	0.108	0.079	**2900.32**	0.145	0.084	0.044
NS-TransE	0.261	336.71	0.447	0.296	0.167	0.156	3948.07	0.437	0.256	0.007
HNSW-TransE	**0.316**	**173.80**	**0.494**	**0.350**	**0.226**	**0.246**	3099.30	**0.539**	**0.407**	**0.050**

First, from the overall perspective of the charts, among the three comparative models, our proposed HNSW-KGE model achieves optimal or near-optimal evaluation results in most cases, particularly on the FB15K237 dataset. Except for the MR evaluation metric on HNSW-ComplEx, HNSW-KGE outperforms all other comparative models across all evaluation metrics. Although in some specific cases, certain comparative models perform slightly better than HNSW-KGE—such as NS-SimplE being marginally better than HNSW-

SimplE on the Hit@10 metric in the WN18RR dataset—these results are close. Moreover, HNSW-SimplE shows average improvements of 2.89% and 6.10% over NS-SimplE on the Hit@1 and Hit@3 metrics, respectively. The experimental results validate the effectiveness of the HNSW-KGE framework.

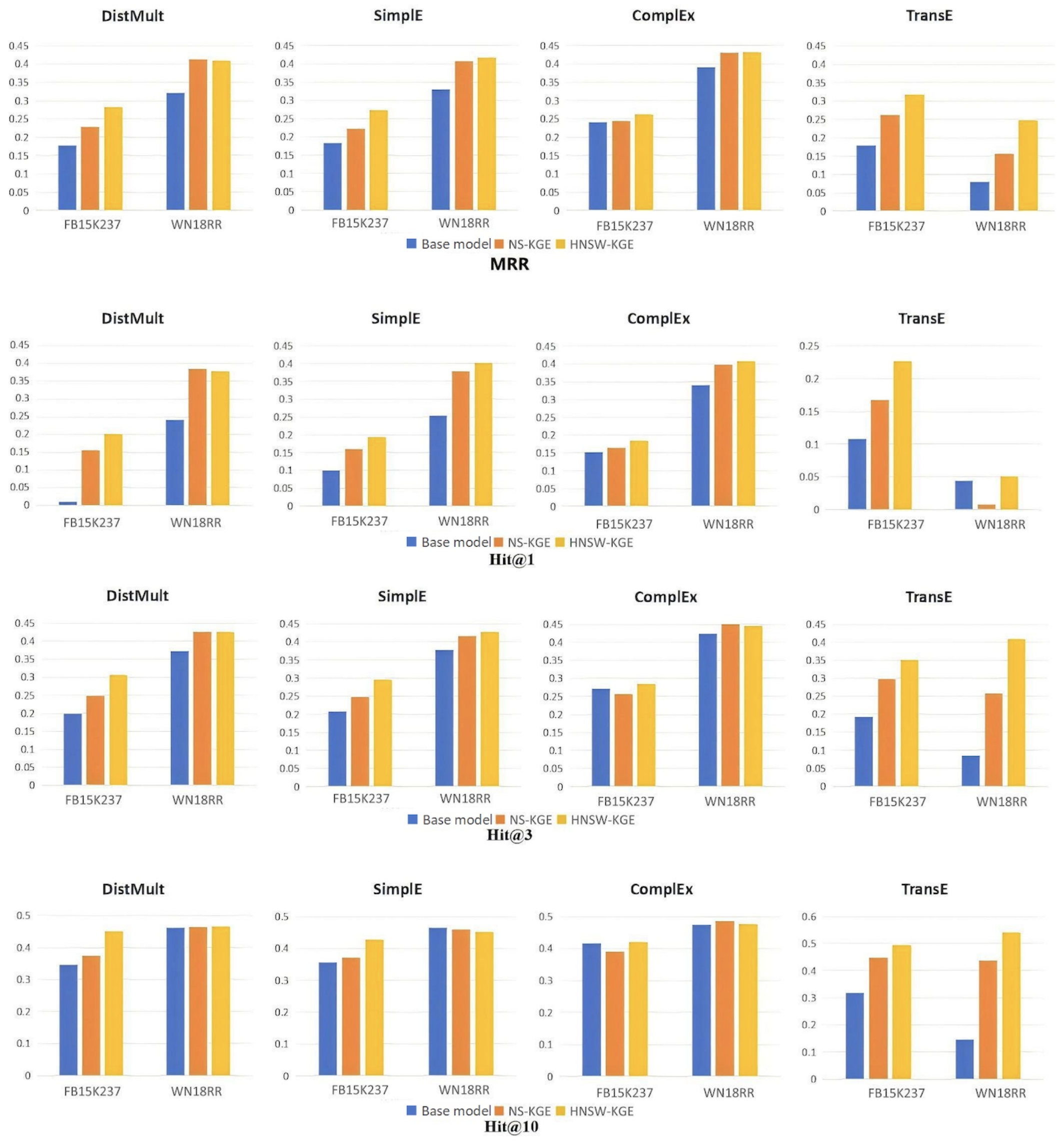

Figure 4. Comparison of evaluation metrics for three types of link prediction methods.

Secondly, compared to the four original baseline models—TransE, DisMult, SimplE, and ComplEx—the models utilizing the HNSW-KGE framework demonstrate an average performance improvement of 58.8% in MRR, 3.2% in MR, 47.2% in Hit@10, 75.1% in Hit@3, and 285.6% in Hit@1 across two datasets. This indicates that the candidate negative sample set constructed using the pre-trained initial embedding vectors and high-quality candidate negative sample set strategy significantly enhances the model training process. Consequently, models employing the HNSW-KGE framework outperform traditional knowledge graph embedding models based on negative sampling.

Furthermore, when compared to the knowledge graph embedding models utilizing the NS-KGE framework, such as NS-TransE, NS-DisMult, NS-SimplE, and NS-ComplEx, the models using the HNSW-KGE framework show an average performance improvement of 16.9% in MRR, 9.9% in MR, 9.2% in Hit@10, 16.6% in Hit@3, and 90.1% in Hit@1 across two datasets. This indicates that, compared to the NS-KGE framework, our proposed HNSW-KGE framework better enhances model training performance. This is because the model training under the NS-KGE framework treats all unobserved samples in the dataset as negative samples, inevitably introducing a large number of false negatives and easy-to-separate negative samples during training. In contrast, the HNSW-KGE framework utilizes relatively high-quality embeddings of entities and relations obtained from pre-training, along with a high-quality candidate negative sample set construction strategy, which effectively reduces the involvement of false negatives and numerous easy-to-separate negative samples in the training process. Consequently, training the model based on these relatively high-quality negative samples yields better results. Additionally, the HNSW-KGE framework incorporates a weight factor based on the frequency of triples into the loss function, allowing long-tail entities and relations to receive more attention during training, thereby mitigating their adverse impact on model training.

Next, compared to the original baseline models and the NS-KGE framework, models using the HNSW-KGE framework consistently achieve better performance in the MRR evaluation metric, although they perform relatively worse in MR, particularly on the WN18RR dataset. This discrepancy arises because MR is more sensitive to prediction errors; if some true entities rank low in the predicted candidate entity list, it significantly affects MR. In contrast, MRR is less influenced due to the reciprocal operation. The superior performance of the HNSW-KGE framework in MRR suggests that, in most cases, it can rank true entities higher. However, the relatively poor performance in MR on the WN18RR dataset is due to the larger number of entities in this dataset compared to FB15K237, making it easier for true entities to rank lower. Finally, the experimental results indicate that the HNSW-KGE framework can enhance the training performance of both semantic matching models (such as DisMult, SimplE, and ComplEx) and translation-based models (like TransE). This demonstrates the framework's universality for training negative sampling models and its potential application to other knowledge graph embedding models.

Finally, the model using the HNSW-KGE framework has the most outstanding average improvement effect on Hit@1, indicating that our HNSW-KGE framework can improve the quality of negative samples involved in training through candidate negative sample set construction and loss function weighting, alleviate the adverse effects of long-tail entities and relations on model training, and thus learn more accurate embeddings for most elements in the knowledge graph so that the model's Hit@1 results in link prediction tasks are significantly improved. In addition, although HNSW-KGE's performance in MR is mediocre, its performance in MRR has basically achieved the best results, indicating that although some real entities rank low in their predicted candidate entity list, making the MR value more sensitive to prediction errors, in most cases, the real entities can be ranked higher. It can be considered that the HNSW-KGE framework can improve the training performance of two types of negative-sampling-based models, such as semantic-matching-based models (DisMult, SimplE, ComplEx) and translation-based models (TransE), showing the versatility of the framework for training models based on negative sampling and its potential for application to other knowledge graph embedding models.

In summary, the comparative experimental results on link prediction tasks indicate that models utilizing the HNSW-KGE framework perform well in this task. Further analysis of the results reveals that the HNSW-KGE framework's approach of using pre-trained embedding vectors and a high-quality candidate negative sample set construction method enables the participation of relatively higher-quality negative samples in training. Moreover, the weighted loss function strategy alleviates the adverse effects of long-tail entities and relations on model training, ultimately leading to better embeddings for knowledge graphs. As a result, models employing this framework benefit significantly in link prediction task experiments.

6.2. Parameter Analysis

Next, this section will explore the impact of the hyperparameter $N \in \{10, 30, 50, 80, 100\}$, which represents the number of candidate negative samples constructed for each positive sample, on the training performance of the HNSW-TransE model using experiments on the FB15K237 dataset. N is tuned on the validation set. It is easier to determine the optimal values of these hyperparameters by performing validation experiments on the validation set. The evaluation metric Hit@K will be used for quantitative analysis, as shown in Figure 5.

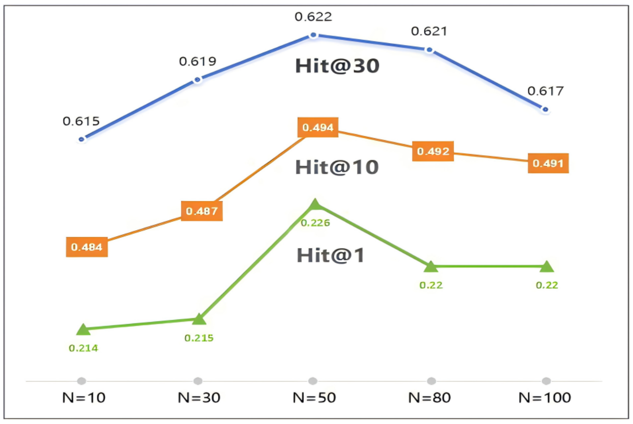

Figure 5. The impact of hyperparameter N on Hit@K in link prediction.

From the figure, it can be observed that, before reaching a specific value of N, the Hit@K values show an increasing trend. However, after reaching that value, as N increases, the Hit@K values start to decline. This demonstrates the effectiveness of N in controlling the quality of negative samples sampled for training. When N is too small, the number of high-quality negative samples sampled is limited. As N increases, the number of high-quality negative samples also increases. Once N reaches a certain value, further increasing N does not yield additional high-quality negative samples. Instead, the increased number of easy-to-separate negative samples can lead to a decrease in model training performance.

The figure also indicates that, during hyperparameter tuning, it is relatively straightforward to identify the optimal number of candidate negative samples N for each positive sample, which is $N = 50$.

6.3. Ablation Analysis

To further illustrate the impact of the high-quality candidate negative sample set construction module and the loss function weighting module on our proposed HNSW-KGE framework, we will present ablation analysis experiments conducted on two datasets: FB15K237 and WN18RR. The experimental results are shown in Table 5, ↓ means the lower value represents better performance. ↑ means the higher value represents better performance.

Table 5. Ablation study results.

Datasets	**FB15K237**					**WN18RR**				
Evaluation Metrics	**MRR ↑**	**MR ↓**	**Hits@10 ↑**	**Hits@3 ↑**	**Hits@10 ↑**	**MRR ↑**	**MR ↓**	**Hits@10 ↑**	**Hits@3 ↑**	**Hits@1 ↑**
HNSW-TransE w/o W_f	<u>0.301</u>	<u>192.7</u>	<u>0.488</u>	<u>0.334</u>	<u>0.211</u>	**0.246**	<u>3239.5</u>	<u>0.533</u>	<u>0.404</u>	<u>0.043</u>
HNSW-TransE w/o HNS	0.300	197.2	0.484	0.332	0.209	0.237	3358.6	0.530	0.403	0.034
HNSW-TransE	**0.316**	**173.8**	**0.494**	**0.350**	**0.226**	**0.246**	**3099.3**	**0.539**	**0.407**	**0.050**

In the table, "HNSW-TransE w/o W_f" indicates that the HNSW-TransE model utilizes only the high-quality candidate negative sample set construction module without the loss function weighting module. Conversely, "HNSW-TransE w/o HNS" indicates that the model uses only the loss function weighting module without the high-quality candidate negative sample set construction module. Bold data in the table represent the best experimental results, while underlined data indicate the second best results.

From the experimental results displayed in the table, it can be observed that using both modules simultaneously in "HNSW-TransE" achieves relatively better evaluation metric values compared to using only one of the modules. This demonstrates that the high-quality candidate negative sample set construction and loss function weighting modules both contribute to the improvement in knowledge graph embedding training performance, indicating a cumulative effect. Additionally, when comparing the results of using only the loss function weighting module to those of using only the high-quality candidate negative sample set construction module, it is evident that the latter yields better knowledge graph embeddings. This highlights that the construction of a high-quality candidate negative sample set contributes more significantly to the model than the loss function weighting module. The key to enhancing the model's training performance lies in selecting high-quality negative samples for training.

Figure 6 also presents the ablation analysis comparison. ↓ means the lower value represents better performance. ↑ means the higher value represents better performance. It shows the comparison of the evaluation indicators of the HNSW-KGE model on the link prediction task without using the high-quality candidate negative sample set construction module. It can be intuitively seen that the high-quality candidate negative sample set construction module improves the model performance. The figure below shows the comparison of the evaluation indicators of the HNSW-KGE model on the link prediction task without using the loss function weighting module. It can be intuitively seen that the loss function weighting module improves the model performance.

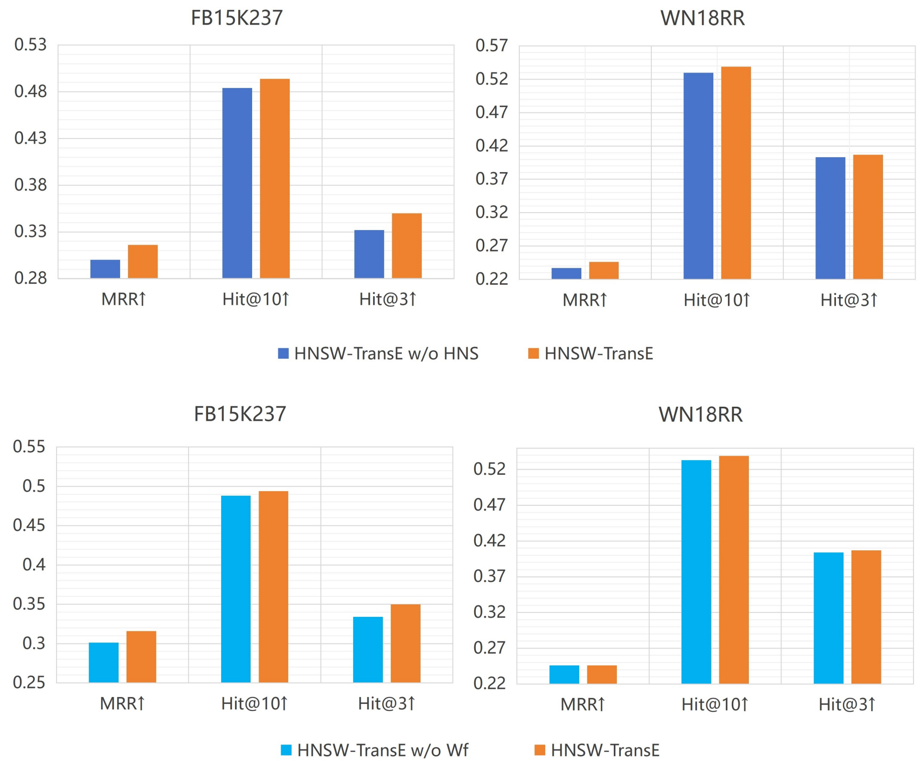

Figure 6. Ablation analysis comparison.

6.4. Case Analysis

Next, we will conduct a qualitative analysis of the entity ranking results in link prediction to intuitively demonstrate the effectiveness of the proposed HNSW-KGE model. We selected the link prediction results of two triples from the experimental outcomes for comparative analysis, as shown in Table 6. Since the NS-KGE framework shows improved performance compared to the original baseline models, we will only select the NS-KGE framework as the comparison method. In the table, HNSW-X refers to the version of model X that applies the HNSW-KGE framework, while NS-X refers to the version of model X that applies the NS-KGE framework. The true entities are highlighted in bold. From the table, it can be observed that, for a given head entity and relation, the true tail entity receives a higher ranking in the model utilizing the HNSW-KGE framework. This indicates that the model applying the HNSW-KGE framework has stronger link prediction capabilities compared to the model using the NS-KGE framework.

Table 6. Link prediction results for two triples.

Datasets	**FB15K237**			
Relation	**/location/imports_and_exports/exported_to**			
Head Entity	**Zambia**		**Angola**	
Model	**HNSW-TransE**	**NS-TransE**	**HNSW-DistMult**	**NS-DistMult**
Top 10 Entities in Link Prediction	**Tanzania**	Namibia	United States of America	New Zealand
	Angola	Botswana	France	Germany
	Zambia	Angola	**People's Republic of China**	India
	France	Zambia	Brazil	United Kingdom
	Sudan	Democratic Republic of the Congo	South Africa	**People's Republic of China**
	Netherlands	Sudan	Portugal	Canada
	Belgium	Malawi	United Kingdom	Australia
	Madagascar	Uganda	Spain	Italy
	United States of America	**Tanzania**	Egypt	United States of America
	Germany	Burundi	Zambia	France

7. Conclusions and Future Work

We proposed a universal knowledge graph embedding framework called HNSW-KGE, which is based on high-quality negative sampling and weighting. First, we utilized the non-sampling-based NS-TransE model for pre-training, allowing for the rapid learning of relatively high-quality embedding vectors for entities and relations in the knowledge graph. Next, we designed a high-quality candidate negative sample set construction strategy that uses the pre-trained embedding vectors to create high-quality candidate negative samples for positive triples, thereby providing high-quality negative samples for training models based on negative sampling. Then, we applied a weight factor based on the frequency of triples to weight the loss function, mitigating the adverse effects of long-tail entities and relations on model training and improving training effectiveness. Finally, to evaluate the performance of the proposed HNSW-KGE, experiments were conducted on four KGE models—DistMult, SimplE, ComplEx, and TransE—confirming the effectiveness of HNSW-KGE.

In the future, we will continue to explore several directions: first, we will investigate other more effective knowledge graph embedding models for pre-training, such as RotatE; second, we will research methods for enhancing model efficiency, such as attempting to use more efficient optimization algorithms to accelerate the candidate negative sample set construction process; third, we will further explore other methods for excluding false negative samples, such as considering the introduction of external knowledge graphs to assist in the identification of false negatives; fourth, we will conduct more tasks to verify the effectiveness of the proposed model, such as question answering or reasoning.

Author Contributions: Conceptualization, P.Z. and H.P.; methodology, H.P. and Y.F.; software, Z.Y.; validation, Y.H. and Z.T.; formal analysis, P.Z. and Z.T.; investigation, W.X.; resources, Y.H. and Y.F.; data curation, Z.Y.; writing—original draft preparation, P.Z.; writing—review and editing, Y.F.; supervision, W.X.; project administration, W.X.; funding acquisition, W.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by National Key R&D Program of China No. 2022YFB3102600, NSFC under grants Nos. 62306322, 72301284 and 72471237, NSSFC uner grant No.2021-SKJJ-C-087.

Data Availability Statement: The datasets can be found at https://github.com/TimDettmers/ConvE [18], accessed on 5 March 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang, F.; Zhang, Z.; Ao, X.; Gao, D.; Zhuang, F.; Wei, Y.; He, Q. Mind the Gap: Cross-Lingual Information Retrieval with Hierarchical Knowledge Enhancement. In Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022, Virtual Event, 22 February–1 March 2022; pp. 4345–4353. [CrossRef]
2. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 3549–3568. [CrossRef]
3. Jia, Z.; Pramanik, S.; Roy, R.S.; Weikum, G. Complex Temporal Question Answering on Knowledge Graphs. In Proceedings of the CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, 1–5 November 2021; pp. 792–802. [CrossRef]
4. Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In Proceedings of the Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, NV, USA, 5–8 December 2013; pp. 2787–2795.
5. Yang, B.; Yih, W.; He, X.; Gao, J.; Deng, L. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
6. Kazemi, S.M.; Poole, D. SimplE Embedding for Link Prediction in Knowledge Graphs. In Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, Montréal, QC, Canada, 3–8 December 2018; pp. 4289–4300.
7. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. Complex Embeddings for Simple Link Prediction. In Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York, NY, USA, 19–24 June 2016; Volume 48, pp. 2071–2080.
8. Wang, P.; Li, S.; Pan, R. Incorporating GAN for Negative Sampling in Knowledge Representation Learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 2005–2012. [CrossRef]
9. Cai, L.; Wang, W.Y. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, LA, USA, 1–6 June 2018; Volume 1, pp. 1470–1480. [CrossRef]
10. Madushanka, T.; Ichise, R. Negative Sampling in Knowledge Graph Representation Learning: A Review. *arXiv* **2024**, arXiv:2402.19195. [CrossRef]
11. Zhang, Z.; Guan, Z.; Zhang, F.; Zhuang, F.; An, Z.; Wang, F.; Xu, Y. Weighted Knowledge Graph Embedding. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, 23–27 July 2023; pp. 867–877. [CrossRef]
12. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge Graph Embedding by Translating on Hyperplanes. In Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 1112–1119.
13. 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, Austin, TX, USA, 25–30 January 2015; pp. 2181–2187.
14. 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 of the Asian Federation of Natural Language Processing, ACL 2015, Beijing, China, 26–31 July 2015; Volume 1, pp. 687–696. [CrossRef]
15. Sun, Z.; Deng, Z.; Nie, J.; Tang, J. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
16. Nickel, M.; Rosasco, L.; Poggio, T.A. Holographic Embeddings of Knowledge Graphs. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 1955–1961.
17. Liu, H.; Wu, Y.; Yang, Y. Analogical Inference for Multi-relational Embeddings. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 2168–2178.
18. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2D Knowledge Graph Embeddings. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 1811–1818.

19. Nguyen, D.Q.; Nguyen, T.D.; Nguyen, D.Q.; Phung, D.Q. 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, NAACL-HLT, New Orleans, LA, USA, 1–6 June 2018; Volume 2, pp. 327–333. [CrossRef]
20. Gardner, M.; Talukdar, P.P.; Krishnamurthy, J.; Mitchell, T.M. Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 25–29 October 2014; pp. 397–406. [CrossRef]
21. Schlichtkrull, M.S.; Kipf, T.N.; Bloem, P.; van den Berg, R.; Titov, I.; Welling, M. Modeling Relational Data with Graph Convolutional Networks. In Proceedings of the Semantic Web—15th International Conference, ESWC 2018, Heraklion, Crete, Greece, 3–7 June 2018; Volume 10843, pp. 593–607. [CrossRef]
22. Zhang, Z.; Zhuang, F.; Zhu, H.; Shi, Z.; Xiong, H.; He, Q. Relational Graph Neural Network with Hierarchical Attention for Knowledge Graph Completion. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020; pp. 9612–9619. [CrossRef]
23. Vashishth, S.; Sanyal, S.; Nitin, V.; Talukdar, P.P. Composition-based Multi-Relational Graph Convolutional Networks. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020.
24. Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Xie, X.; Guo, M. GraphGAN: Graph Representation Learning With Generative Adversarial Nets. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; pp. 2508–2515. [CrossRef]
25. Hu, K.; Liu, H.; Hao, T. A Knowledge Selective Adversarial Network for Link Prediction in Knowledge Graph. In Proceedings of the Natural Language Processing and Chinese Computing—8th CCF International Conference, NLPCC 2019, Dunhuang, China, 9–14 October 2019; Volume 11838, pp. 171–183. [CrossRef]
26. Zia, T.; Windridge, D. A generative adversarial network for single and multi-hop distributional knowledge base completion. *Neurocomputing* **2021**, *461*, 543–551. [CrossRef]
27. Tang, C.; Rao, Y.; Yu, H.; Sun, L.; Cheng, J.; Wang, Y. Improving Knowledge Graph Completion Using Soft Rules and Adversarial Learning. *Chin. J. Electron.* **2021**, *30*, 623–633.
28. Liu, L.; Zeng, J.; Zheng, X. Learning structured embeddings of knowledge graphs with generative adversarial framework. *Expert Syst. Appl.* **2022**, *204*, 117361. [CrossRef]
29. Lei, J.; Ouyang, D.; Liu, Y. Adversarial Knowledge Representation Learning Without External Model. *IEEE Access* **2019**, *7*, 3512–3524. [CrossRef]
30. Krompaß, D.; Baier, S.; Tresp, V. Type-Constrained Representation Learning in Knowledge Graphs. In Proceedings of the Semantic Web—ISWC 2015—14th International Semantic Web Conference, Bethlehem, PA, USA, 11–15 October 2015; Volume 9366, pp. 640–655. [CrossRef]
31. Weyns, M.; Bonte, P.; Steenwinckel, B.; Turck, F.D.; Ongenae, F. Conditional Constraints for Knowledge Graph Embeddings. In Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG2020) co-located with the 17th Extended Semantic Web Conference 2020 (ESWC 2020), Heraklion, Greece, 2 June 2020; Volume 2635.
32. Jain, N.; Tran, T.; Gad-Elrab, M.H.; Stepanova, D. Improving Knowledge Graph Embeddings with Ontological Reasoning. In Proceedings of the Semantic Web-ISWC 2021—20th International Semantic Web Conference, ISWC 2021, Virtual Event, 24–28 October 2021; Volume 12922, pp. 410–426. [CrossRef]
33. Niu, G.; Li, B.; Zhang, Y.; Pu, S. CAKE: A Scalable Commonsense-Aware Framework For Multi-View Knowledge Graph Completion. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, 22–27 May 2022; pp. 2867–2877. [CrossRef]
34. Hajimoradlou, A.; Kazemi, M. Stay Positive: Knowledge Graph Embedding Without Negative Sampling. *arXiv* **2022**, arXiv:2201.02661. [CrossRef]
35. Bahaj, A.; Lhazmir, S.; Ghogho, M. KG-NSF: Knowledge Graph Completion with a Negative-Sample-Free Approach. *arXiv* **2022**, arXiv:2207.14617. [CrossRef]
36. Li, Z.; Ji, J.; Fu, Z.; Ge, Y.; Xu, S.; Chen, C.; Zhang, Y. Efficient Non-Sampling Knowledge Graph Embedding. In Proceedings of the WWW '21: The Web Conference 2021, Virtual Event, Ljubljana, Slovenia, 19–23 April 2021; pp. 1727–1736. [CrossRef]