Journal of Computer and Knowledge Engineering, Vol. 5, No. 2, 2022. (11-20)
DOI: 10.22067/cke.2022.73802.1038

Embedding Knowledge Graph through Triple Base Neural Network and Positive Samples*

Research Article

Sogol Haghani[1] Mohammad Reza Keyvanpour[2]

Abstract: Representation learning on a knowledge graph aims to capture patterns in the knowledge graph as low-dimensional dense distributed representation vectors in the continuous semantic space, which is a powerful technique for predicting missing links in knowledge bases. The problem of knowledge base completion can be viewed as predicting new triples based on the existing ones. One of the prominent approaches in knowledge base completion is the embedding model. Currently, the majority of existing knowledge graph embedding models cannot deal with unbalanced entities and relations. In this paper, a new embedding model is proposed, with a general solution instead of using the additional corpus. First, a triple-based neural network is presented to maximize the likelihood of the knowledge bases finding a low-dimensional embedding space. Second, two procedures to generate positive triples are proposed. They produce positive triples and add them to the training data. The policies can capture rare triples, and simultaneously remain efficient to compute. Experiments show that the embedded model proposed in this paper has superior performance.

Keywords: Knowledge Graphs, Link Prediction, Positive Samples, Embedding Neural Network, Graph Mining

1. Introduction

Knowledge bases like Wordnet [1], YAGO [2], or the Google Knowledge Graph are useful resources used in many AI tasks, which present ways to organize, manage, and retrieve all digital knowledge. A knowledge base can be represented as a set of (head, relation, and tail) triples. Any information can reach from the knowledge base through triples or concatenation of them [3, 4]. Although completeness, accuracy, and high quality of data are the parameters that guarantee their advantage of them, they suffer from incompleteness and a lack of reasoning capability [3]. The problem of knowledge base completion can be viewed as predicting new triples based on the existing ones [6].

One of the promising approaches to knowledge base completion is to embed their entities and relations into low-dimensional vector spaces. The methods define a score function and assign a score to the triple [5, 6]. For any unobserved triple, its plausibility can be predicted by using the learned embedding and the score function. The high-value score will assign to the probable triple [5].

Despite the substantial efforts and great successes in the research, the effectiveness of the embedding methods has not been directly compared. They mostly use various pre-training methods to initialize the embedding vector space. It is still unclear that which pre-training method should be employed, though it has a considerable effect on the results [7, 8]. Another issue is heterogeneous and unbalanced entities and relations in the knowledge base. Heterogeneity may affect overfitting on simple triples or underfitting on rare ones. A simple triple is the one in which its elements appear in most other triples, while rare triples lack their entities and relations of looking most [9]. In Fig.1 triple $(F, live\_in, E)$ is such a rare triple that the rate of relation $live\_in$ is lower than the other, or triple $(G, father\_of, H)$ is the other kind rare one, which the degree of H is low in comparison to G. Alternatively, triple $(F, born\_in, D)$ is such a simple one. Although embedding methods have a strong ability to model knowledge graphs, it remains challenging faced with heterogeneous data [10].

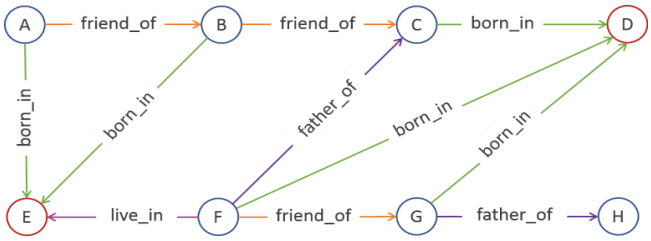

Figure 1. Example of rare and simple triple

The goal of this study is to introduce a novel algorithm that does not require pre-training and can perform and compete while it can deal with unbalanced entities and relations. To that end, two methods are proposed.

First, we propose a new triple-based embedding neural network, to encode the knowledge base to the embedding vector space for entities and relations which maximizes the likelihood of the whole knowledge base. It is a customized, objective function using Stochastic Gradient Descent (SGD) motivated by prior work on natural language processing to the triple structure [11]. The proposed triple-based embedding neural network was used to capture the semantic and syntactic structure of the knowledge base. It takes a knowledge graph as input and produces latent representations for entities and relations. On this subject, we showed that the triple-based embedding neural network used in knowledge base completion obtains proper results in comparison to the state of the arts.

Second, since the embedding models lack in predicting rare triples, two different procedures are introduced to augment the knowledge base to overcome this deficiency. To address this issue, positive triples are generated during the training with a semi-learned embedding vector. Generated triples are added to the training data based on the rate of appearing in previous training data. The rarer triple, the

* Manuscript received: 22 November 2021, Revised, 07 July 2022; Accepted: 03 October 2022.
[1]. Master, Department of Computer Engineering, Alzahra University, Tehran, Iran.
[2]. Corresponding author. Professor, Department of Computer Engineering, Alzahra University, Tehran, Iran.
Email: Keyvanpour@alzahra.ac.ir

higher the chance of being in the training set. Each procedure uses a specific mechanism in adding positive triples to the training data.

- GNSs (Generate New Samples) generate positive triples after δ iteration of learning triple-based embedding neural network, then add them to the training set. The rest of the iterations are worked with the new augmented training set.
- FCSA (Flip Coin Simulated Annealing) decides to generate new triples or use the training sample in the learning process. At the beginning of the process, it rarely generates new triples, and by the time when the embedding vectors learn, it can generate more new triples.

We demonstrate their usefulness by applying them to our triple based neural network. Our extensive experiments on two benchmark datasets show that they achieve superior performance over competitive baselines in two knowledge base completion tasks.

The rest of the paper is structured as follows. Section 2 reviews literature on knowledge base embedding. Section 3 presents our approach. Section 4 presents empirical results. Finally, section 5 includes the conclusion and plan of further work.

2. Related works

Various models have been proposed for knowledge graph completion through the link prediction task. Embedding the knowledge graph into a low-dimensional continuous vector space is one of the assuring approaches [12]. Various types of knowledge graph embedding models have been proposed, and they learn the relation between entities using observed triple in the knowledge graph. These models can be classified into three classes: translation-based models, bilinear models, and compositional models [6]. Before proceeding, mathematical notations need to be defined. h, r, and t denote a head entity, relation, and tail entity, respectively.

The bold letters e_h, e_r , and e_t denote embeddings of h, r, and t, respectively, on an embedding space $\mathbb{R}^d$. E and R represent sets of entities and relations, respectively.

Translation-based models

The existing translation-based model treats the triple as a relation-specific translation from the head entity to the tail entity. The entity vector obtains the optimal value during the training process by score function, while the relation is regarded as an operator or a translator [5, 12]. Meanwhile, TransE has been introduced as a pioneer in this approach [13]. It is assumed that there is $e_h + e_r \approx e_t$ equation for each valid triple which assumes that the tail embedding e_t should be in the neighborhood of $e_h + e_r$. TransE is used L_2 to learn embedding vectors. It is not only a simple model but also has a high degree of scalability for modeling complex patterns by embedding dimensions. TransH [10], TransD [9], and TransR [14] are other translation methods. For instance, TransH is a transitional projection. TransD is similar to it, with the difference that it uses the identity matrix of $d \times k$ size. The dimensionality of the entity and relation vector is considered differently. TransR also uses a rotation transformation for the train. CTransR [14] and TransSparse [9] are an extension of TransR. CTransR considers correlations under each relation type by clustering diverse head-tail pairs into groups and learning distinct relation vectors for each group. TransSparse focuses on solving the imbalance issues in knowledge graphs, which are ignored by previous translation models. The imbalance means that the number of head entities and that of tail entities in relation could be different.

Bilinear models

The DistMult [15] is based on a bilinear model where each relation is represented by a diagonal rather than a full matrix. It learns a tensor that is symmetric in the subject and object, while datasets contain mostly non-symmetric triples. ComplEx [12] solves the same issue of DistMult by the idea that multiplication of complex values is not symmetric. ComplEx represents a real-valued tensor $X \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ as the real part of the sum of *R* complex-valued rank one tensors $u_r^{(1)} \otimes u_r^{(2)} \otimes \overline{u_r^{(1)}}$ where $r \in \{1, \dots, R\}$ and $u_r^{(m)} \in C^{N_m}$

$$f_r(h,t) = Re(\sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes \overline{u_r^{(1)}}) \quad (6)$$

where $\overline{u_r^{(1)}}$ is the complex conjugate of $u_r^{(1)}$. Bilinear models have more redundancy than translation-based models and so easily become overfitted. Hence, embedding spaces are limited to low-dimensional space. SimplE [34] are all proved to be fully expressive when embedding dimensions fulfill some requirements. The full expressiveness means these models can express all the ground truth which exists in the data, including complex relations. However, these requirements are hardly fulfilled in practical use. RotatE [35] represents relations as rotations in a complex latent space, with h, r, and t all belonging to C^d. The r embedding is a rotation vector: in all its elements, the complex component conveys the rotation along that axis, whereas the real component is always equal to 1. The rotation *r* is applied to h by operating an element-wise product (once again noted with $\odot$ in 1). L1 norm is used for measuring the distance from t. The authors demonstrate that rotation allows modeling correctly numerous relational patterns, such as symmetry/anti-symmetry, inversion, and composition.

Compositional models

In the LP field, KG embeddings are usually learned jointly with the weights and biases of the layers; these shared parameters make these models more expressive, but potentially heavier, harder to train, and more prone to overfitting [33]. NTN [16] is one of the most well-known methods in knowledge base completion. The model uses a three-way tensor in its score function. In other words, NTN can replace the standard neural network layer with a three-way tensor layer. Also, using $tanh$ for applying the non-linear actions, the score function can be calculated as follows:

$$f_r(h,t) = u_r^{\mathrm{T}} f(e_h^{\mathrm{T}} \underline{W_r}^{[1:k]} e_t + W_{r,1} e_h + W_{r,2} e_t + b_r) \quad (7)$$

where $\underline{W_r}^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor and $W_{r,1}, W_{r,2} \in \mathbb{R}^{k \times d}$ are weight matrices and $b_r \in \mathbb{R}^k$ is the bias vector. Despite the fascinating performance, this method is very complicated, and the evaluation results show that representations vectors with the pre-train can reach such a function [17].

HOLE [18] is another method known in this field. This

method has high performance compared to the others. The reason for this function is that it can be applied to a circle of correlation in the score function to represent the space of entities and relations. This method uses a pre-train to create the initial representation space, which causes representation vectors to not have random values at the beginning of the training process and, conversely, have an appropriate initialization.

ConvE [31] performs a global 2D convolution operation on the subject entity and relation embedding vectors after they are reshaped to matrices and concatenated. The obtained feature maps are flattened and transformed through a linear layer, and the inner product is taken with all object entity vectors to generate a score for each triple. Whilst results achieved by ConvE are impressive, the reshaping and concatenating of vectors as well as using 2D convolution on word embeddings is unintuitive. The R-GCN uses a graph convolutional network to obtain an embedding of the triples, then applies DistMult [15] to compute a score for the embeddings.

As pointed out in [8], pre-training is an open question where it is still unclear which pre-training method should be employed. There is no standard, and no priority has been mentioned for it.

3. Our approach

In this section, we first propose how the triple-based embedding neural network is worked to represent entities and relations. Second, the detail of generating positive triples and two procedures of how to apply them in learning is provided.

3.1. Triple-Based Embedding Neural Network

Figure 2 shows a perspective of the Triple-based Embedding Neural Network's layers. It consists of three layers. As seen in the figure, the first layer is composed of two parts connected by the weight matrices to the hidden layer. The upper part of the layer is a one-hot vector of the head entity, and the bottom is a one-hot vector of the relation. The hidden layer is a sum of the projection vectors of head and relation. The number of neurons in the last layer is also equal to |E|, which is equal to the size of the upper part of the first layer. This layer describes the probability of tail with the given of the head and relation. In other words, not only the last layer is not the output but also the embedding vectors are its rows of weight matrices.

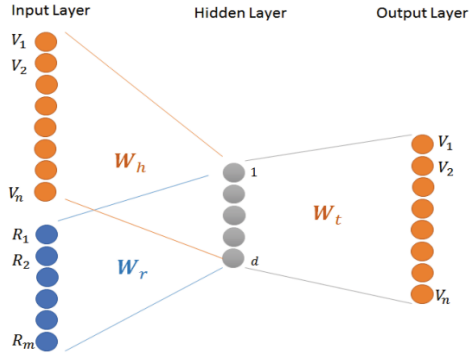

Figure 2. Triple Based Embedding Neural Network

Three weight matrices W_h, W_r and W_t after training have optimal weights, and each rows of W_h^{T}, W_t and W_r^{T} are a embedding vector for entities e_h, e_t and relation e_r [11, 13]. The overall process of learning embedded neural network has been presented in algorithm 1.

Algorithm 1: Triple based Embedding Neural Network

Input: T: Triples, E: Entities, R: Relations, kns: k negative samples
Output: $\boldsymbol{W_h}, \boldsymbol{W_r}, \boldsymbol{W_t}$
Initializations: $\boldsymbol{W_h}$ uniform for each entity as head, $head \in E$, $\boldsymbol{W_h}$ uniform for each entity as tail , $tail \in E$, $\boldsymbol{W_r}$ uniform for relation, $relation \in R$
Create Unigram Table of Negative Triples
foreach n do
  foreach triple do
    Find K negative samples
    Calculate: $log(1/1+e^{-z}) + log(1/1+e^{z})$
    Update $\boldsymbol{W_h}, \boldsymbol{W_r}, \boldsymbol{W_t}$, use SGD.
  end
end

The purpose of the Triple-based embedding neural network is to estimate the maximum likelihood of a knowledge base. Accordingly, as shown in algorithm 1 the main loop of learning tries to maximize its likelihood by considering all training triples of the knowledge base. A loss function should minimize the error by considering corrupted triples [3].

It should be noted that the purpose of the method is to learn latent representations, not probable distribution between two entities. Conditional probability $Pr(t|h,r)$ is considered for triple (h,r,t). The goal is to set the parameter θ to maximize the probability of the knowledge base (8).

$$\arg\max_{\theta} \prod_{triple \in T} Pr(t|h,r;\ \theta) \tag{8}$$

T is the list of observed triples or training sets. $Pr(T = 1|(h,r,t))$ is the probability that the triple (h,r,t) exists in the training set, or, more precisely, a triple has been observed.

Conversely, the probability of $Pr(T = 0|(h,r,t)) = 1 - Pr(T = 1|(h,r,t))$ indicates that a triple has not been observed. With these assumptions, the goal is to find the parameters that maximize the likelihood of seeing all the observed triples in the training set:

$$\begin{aligned} &\arg\max_{\theta} \prod_{triple \in T} Pr(T = 1|(h,r,t);\ \theta) \\ &\approx \arg\max_{\theta} log \prod_{triple \in T} Pr(T = 1|(h,r,t);\ \theta) \\ &= \arg\max_{\theta} \sum_{triple \in T} log\ Pr(T = 1|(h,r,t);\ \theta) \end{aligned} \tag{9}$$

The sigmoid function is used to determine the value of $Pr(T = 1|(h,r,t);\ \theta)$, which is defined as:

$$Pr(T = 1|(h,r,t);\ \theta) = \frac{1}{1+e^{-z}} \tag{10}$$

and it is expected to meet the objective shown in the

formula 11 [11].

$$\arg\max_{\theta} \sum_{triple \in T} log = \frac{1}{1+e^{-z}} \quad (11)$$

To the triple based embedded neural network structure, the parameter z is defined as follows:

$$z = (e_h + e_r).e_t \quad (12)$$

e_h , e_r, and e_t are embedded vectors. They are for $head$, $relation$, and $tail$ respectively. These are the rows of W_h^{T}, W_r and W_r^{T} weight matrices. Figure 3**Error! Reference source not found.**. illustrates the explanation of the equation 12 in vector space. According to the cosine similarity, the smaller the angle between $e_h + e_r$ and $\boldsymbol{e_t}$, maximize the dot product [13]. Due to Figure 3**Error! Reference source not found.**. it is desirable that the sum of $\boldsymbol{e_h}$ and $\boldsymbol{e_r}$ be parallel with e_t [11].

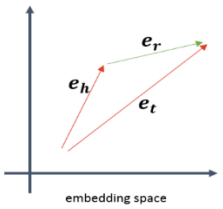

Figure 3. An overview of the relation between e_h , e_r, and e_t vectors

Due to the structure of Triple-based embedding neural network, corrupted triples are used in learning. The table of corrupted triples with uniform distribution is created []. As shown in 1 the table is generated before the main loop. The relevant question is, can a corrupted generated triple be an observed one. In response, it should be stated that there is no claim to the injection of noise in the learning procedures. Owing to the high dimensionality of entities and their relations, the probability of being a missing triple is low [11]. Finally, gradient descent is used to update the weights. As shown in Algorithm 1, all weight matrices are randomly initialized. By the continuation of the training, optimized weights are obtained.

TransE is one of the popular models on large datasets due to its scalability. Similar to TransE, the time complexity of Triple based neural network is $O(d)$, where d is the size of embedding vectors, it is more efficient than ConvE, NTN, and the neural network models [4].

3.2. Generate positive triples

In this section, we start by explaining why to generate positive triples and then describe how to construct them. In the next two sections, the two distinct procedures of how to apply them in the learning model will be illustrated.

Triples are highly heterogeneous in knowledge bases [5]. The diversity is evident both in the type of relation and in the entities. Most of the presented embedding methods are incapable of dealing with such heterogeneity [9]. Therefore, rare entities and relations get an argument. We try to augment rare ones to get a consistent knowledge base. To the best of our knowledge, there has not been an attempt to petition to gain consistent a knowledge base. Inspired by machine vision, data augmentation is used to imbalance classification. Hence, it is being tried to create new images from existing ones and add to the unbalanced classes [20, 21]. Such a mechanism is needed to balance the knowledge base, though creating new triples from existing ones is not possible in this manner.

To address this problem, we adopted the idea of sequence modeling which is stated that the learning model randomly predicts the next sequence at first, and with learning, the model can correctly predict the following one [22]. In these circumstances, the triple-based embedding neural network is allowed to be learned: the model can generate new triples even as the weight matrices are updating. In other words, after several repetitions, the embedding vectors were found to have reasonably optimized: they were able to predict new instances.

For each entity, all possible triples are created, which it has located as head or tail, and the probability of being a true triple is calculated. Then N top of the probable triples is nominated to be used in the learning model. These candidates are chosen concerning their rareness: the rarer relation and entity, the more chance to be selected. In other words, a triple has a higher chance of being selected when the head, tail, or relation has been less commonly observed in the training set. The pseudo-code on how to Generate Positive Triples has been shown in Algorithm 2. In the following sections, two strategies named GNSs and FCSA describe explaining how to use new triples in the learning model.

Algorithm 2: Generate Positive Triples

```
Generate Positive Triples (V, T, R, λ, θ, W_h^T, W_r , W_r^T);
Input: V: entities, T: Triples (training data), R: Relations, λ: Size of the selected
entity, θ: Threshold
Output: Positive Triples
for i = 0 to λ do
    n1 ← Select a node with respect to reverse degree of the node
    foreach relation as r in Relations do
        foreach node as n2 in entities do
            calculate probability for (n1, r, n2) and (n2, r, n1)
            if probability > θ and triple not in Triples then
                temp ← triple
            end
        end
    end
    Select n triples from temp with respect to reverse repetition of relations and
    add to the Triples
end
```

A. GNSs

Figure 4 shows the whole process of when to apply GNSs. In the GNSs strategy, the learning procedure stops after δ repetitions, and the model starts generating new positive triples. These are created by the updated weights matrices and then add to the training set. Then, the learning model continues training with a new training set. In other words, the new set has the original triples and the new positive triples, which predicts by the semi-learned model. Entities and relations in which there is a higher chance of prediction regarding node reverse degree $\frac{1}{\deg(entity)}$ and relation repetition $\frac{1}{|\text{relation}|}$ can benefit from the algorithm. The more infrequent relation and entity, the more chance to predict. In other words, a triple has a higher chance of being selected when $head$, $tail$ or $relation$ has been less commonly observed in the training set. In the opinion of the results of the experiments, selecting a part of the probable triples will increase the performance of the method. According to a thumb rule, the size of the new samples should not be in such a way that eliminates the effect of the original samples.

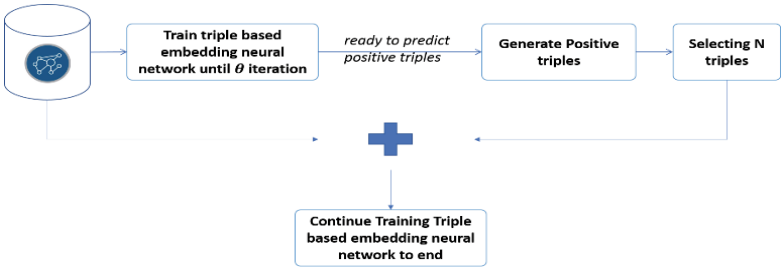

Figure 4. An overview of applying GNSs strategy

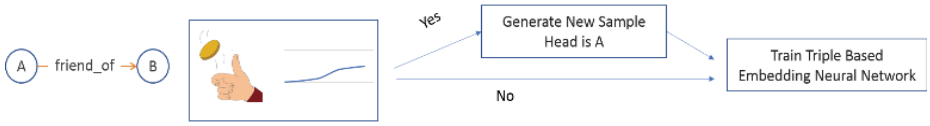

Figure 5. An overview of applying FCSA strategy

The time complexity of finding positive triple is $O(d)$, and it repeated to z times where $z \ll |training\ data|$. As the size of generated positive triples is much less than the training data and they are calculated once in the training, it remains efficient to apply this procedure to the learning.

B. FCSA

The fundamental idea of FCSA is illustrated in Figure 5. During training, FCSA decides whether a new true triple will be generated or use the original one. In contrast with GNS, the training procedure never pauses. For every sample, we propose to flip a coin and use the true triple or generate the probable one. At the beginning of training, sampling from the model would yield a random triple since the model is not well trained. So, selecting more often, the original samples should help. We thus propose to use a schedule to help the model to generate new triples when it becomes more learned. A sigmoid function is used to decide when new triples can be generated:

$$\epsilon = \frac{1}{1+e^{z}} \tag{13}$$

$$z = total\ iteration - current\ iterration$$

It states that the chances of choosing a new triple are higher at the closure of the learning process and expects the model to sample reasonable triples [22]. If a new sample is selected, FCSA will replace the original. FCSA's goal is to explore newer spaces. As in GNSs, the greater chance is given to probable triples which are more heterogeneous when selecting alternative triples. As the size of training sets remains constant the time complexity of FCSA is $O(d)$.

4. Experiments

This section proposes an experimental comparison of the proposed method and demonstrates that it can compete with current state-of-the-art methods [3, 18]. The evaluations are based on Wordnet11 and Wordnet18.

4.1. Datasets and metrics

To evaluate the proposed method, two datasets Wordnet11 [16] and Wordnet18 [13] were used: both are state-of-the-art methods. The statistics of these data sets are given in Table 1.

Wordnet11 and Wordnet18 are not only different from each other regarding the size of entities and relations, but also in the structure of the test and the validation set. Each dataset and assessment criteria are described individually in the following sections.

- Wordnet11: Positive and negative samples are indicated in the triple format with a label in test and validation sets. In other words, triples with negative and positive labels are wrong and right triples respectively. Negative triples are constructed from the corruption of positive ones.

Test methodology

Due to the structure of the dataset, link prediction became a binary classification issue. For each relation, a threshold θ_r was determined for evaluation by the validation set. Therefore, the probability of each triple in the test set was compared with its relation threshold: this determined the decision to put a positive or negative label [16].

Evaluation criteria

Accuracy is a criterion for evaluating this data set, as shown in Equation 14 [23].

$$\text{Accuracy} = \frac{tp+tn}{tp+tn+fp+fn} \tag{14}$$

- Wordnet18: In this dataset, all triples were positive. Therefore, the test methodology and evaluation criteria were based on the triple's rank.

Test methodology

The rank of the triple was calculated following what is mentioned in [13]. Accordingly, for each examination sample, the tail of the triple was replaced with all entities, and the probability for each of them was calculated. The same procedure is also applied to the head entity. Finally, two lists of all created triples were sorted in descending order by their probability. This procedure is called raw mode, which is composed of all possible triples. Another mode is called filtered, in which all created triples that exist in the training, test, and validation sets are removed except the one that should be evaluated [13].

Table 1. Statistics of the experimental datasets used in this study (and previous works). #Entity is the number of entities, #Relation is the number of relation types, and #Train, #Validation and #Test are the numbers of triples in the training, validation and test sets, respectively

Datasets	#Entity	#Relation	#Train	#Validation	#Test
Wordnet11	38,696	11	112,581	2,609	10,544
Wordnet18	40,943	18	141,442	5,000	5,000

Evaluation criteria

MR, MRR, and $Hit@k$ are the evaluation criteria used for Wordnet18. The mean of the triple's rank is called the mean rank MR. MR is in the range of $[\,1\,,\ \infty\,)$. As MR gets close to 1, it shows that the proposed method can predict triples at lower ranks [5] which indicates the efficiency of the method.

$$MR = \frac{\sum rank_i}{|N|} \tag{15}$$

The Mean Reciprocal Rank (MRR) is a statistical measure for evaluating each process that presents a list of possible responses to a sample of questions that are arranged with the correct probability. After calculating the rank of all triples, the MRR is calculated as follows:

$$MRR = \frac{1}{|N|}\sum_{i=1}^{|N|}\frac{1}{rank_i} \tag{16}$$

MRR is in the range of $[0, 1]$. $Hit@k$, like the mean rank criterion, is used to evaluate the prediction of links in the knowledge base. The triple is considered as predicted when the rank is less or equal to K. Finally, the ratio of predicted triples to the total has been shown as the criterion of $Hit@k$ (17).

$$Hit@k = \frac{Number\ of\ Triples\ that\ ranks\ less\ or\ equal\ than\ K}{Number\ of\ All\ Triples} \tag{17}$$

$Hit@k$ is in the range of $[0, 1]$. As the value of this criterion is higher, it shows that most of the triples get a rank lower or equal to k [18].

4.2. Experimental setup

In training the triple-based embedding neural network, two learning rates α and β are used for entity and relation respectively. The learning rate is validated in $\{0.001, 0.01, 0.03, 0.05, 0.07, 0.1, 0.15\}$ and the learning rate α are 0.03 and 0.07 in Wordnet11 and Wordnet18 respectively. Also, the learning rate β, among the values $\{0.001, 0.005, 0.01, 0.05, 0.1\}$, 0.005, and 0.001 is validated for Wordnet11 and Wordnet18 respectively. The reason for adding the β learning rate is that the error calculated from the model is added to the relationship weights at a different rate. Since the entity-to-relation ratio is very heterogeneous, two learning parameters are needed to tune the neural network. The appropriate number of negative samples for learning triple base neural network is considered $kns = 1$ and $kns = 5$ in Wordnet11 and Wordnet18 [16]. Furthermore, the number of positive triples in the training process in the GNSs strategy is estimated at 500 and 1500 in Wordnet11 and Wordnet18, respectively. By increasing large numbers of positive triples noise can spread. While, the less samples impact minor effect on results. In GNSs the δ is equal to $3/4$ total iteration for each data set.

4.3. Baselines

This paper compared several state-of-the art relational learning approaches. TransE, TransR, R-GCN, NTN, ComplEx, ConvE and R-GCN comprise our baselines. The results of TransE, R-GCN, TranSparse-DT, and ComplEx are reported from [12] and the results of TransR and NTN from [36], and the rest are from [31]. They are current, state-of-the-art methods and they use the same evaluation protocol.

4.4. Results

To specify the effect of each method, four distinct examinations are presented:

1. ENN: Train Triple-based Embedding Neural Network Without Any Strategy;
2. ENN + GNSs: Train Triple-based Embedding Neural Network with GNSs;
3. ENN + FCSA: Train Triple-based Embedding Neural Network with FCSA;
4. ENN + GNSs + FCSA: Train Triple-based Embedding Neural Network with both GNSs and FCSA strategies

4.4. Results on Wordnet11

The results of the four examinations are provided in Figure 6. To illustrate the different aspects of the neural network's capabilities and proposed strategies, these examinations are presented. We also consider the results by the label of relation, classifying each relation according to its labels. It can be seen from Figure 6. that ENN detects their accuracy less than others, such as the *domain topic* and the *domain region*, by applying the strategies, the accuracy of each has increased about 7%. Also, the relations chart shows that the amount of heterogeneity of the relations causes the strategies to have an effect on the accuracy of each relation. For instance, the *synset domain topic* relation that the ENN estimates its accuracy more than *domain topic* and *domain region*, with applying the strategies the results show less improvement compared the two mentioned. Even in some relations, there is no increase in accuracy. In *member holonym* and *member meronym* relations, the accuracy of the ENN is greater than applying strategies (these relations have the highest accuracy among them). The difference is about *0.5%*. This phenomenon shows the decreasing effect of original samples or existence noise in applying strategies. However, it is worth noting that such decreasing is negligible in comparison with the increase of accuracy in other relations.

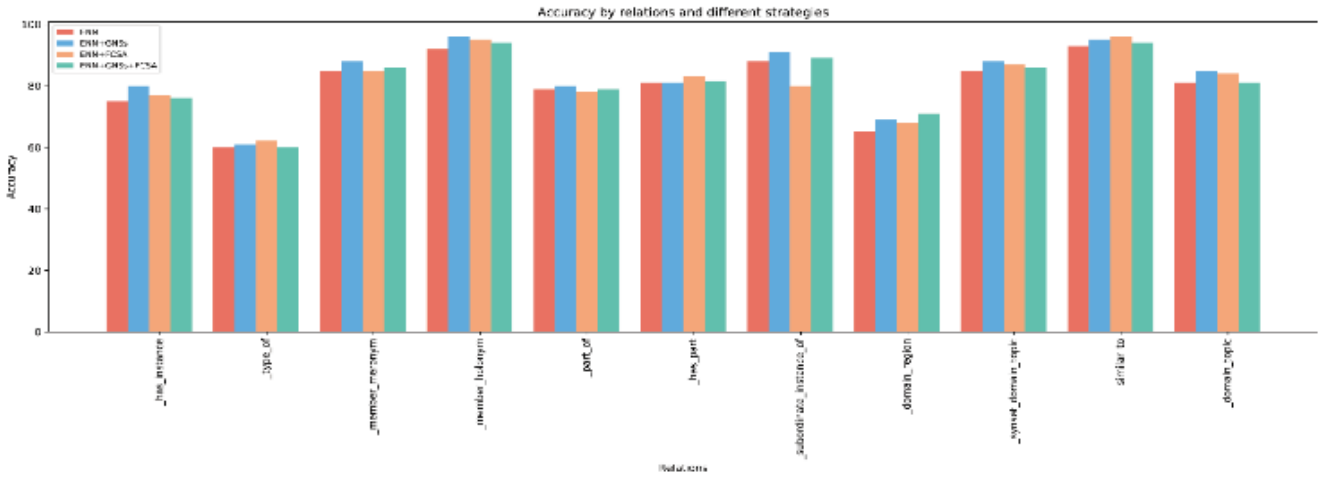

Figure 6. Accuracy of each relation with 4 different tests

The results demonstrate that ENN associated with GNSs is more accurate in comparison with ENN + FCSA. Not only ENN + GNSs consider the training set at the end but also a new training set has been given to the triple-based embedding neural network. Comparing the superiority of the applying FCSA strategy is to reduce the error in high-frequency relations such as *member holonym* and *member meronym*. In this regard, it can be ensured that the accuracy of applying FCSA is not as good a less adverse effect, and is relatively more stable. The ENN has an accuracy of 87.3% and by applying strategies GNSs and FCSA, the accuracy increases about 1 and 2 percent, and this demonstrates that the proposed strategies have a positive effect on the performance of the method. Applying strategies at the same time performs inversely and does not increase accuracy. The cause of this deterioration is related to various aspects. First, by performing FCSA, choosing new samples occurs more when the model is close to the end of the training. If applying FCSA occurs with GNSs, it is probable that some of the new positive samples generated by GNSs will be changed again with FCSA and will be reduced the effect of the GNSs strategy. Also, simultaneously applying these two strategies will cause the original samples at the end of training more faded, and the actual samples do not have their effect [24, 25].

In table 2, all four our distinct examination accuracy with the previously reported results on Wordnet11 are compared. Besides their accuracy, the optimization function that they use for pre-training is shown. Some models have used optimization functions to avoid overfitting. For instance, the NTN method achieves the accuracy of 70.6 without any pre-training, while initializing the embeddings with an unsupervised semantic word vector the accuracy increases to 86.6. Table 2 shows the same result for TransE. Pre-training is used to prevent overfitting, mainly on simple relations. Each model uses distinct methodologies, which makes the comparison not reasonably fair. However, as pointed out by [10] and [34], averaging the pre-trained word vectors for initializing entity vectors is an open problem, and it is not always beneficial since entity names in many domain-specific knowledge bases are not lexically meaningful. However, a comparison has not been made on their performance independently.

According to Table 2, ENN has a high accuracy compared to methods with the same conditions (without any optimization). It shows that the triple-based embedding neural network is robust to overfitting. Also, applying the GNSs strategy has the highest efficiency among all previous states of the arts. It does not only increase the performance but also it is not domain-specific and does not need external data.

Table 1. Link prediction results on Wordnet11

Methods	Acc%	Opt
NTN [16]	70.06	None
NTN [16]	86.2	Initiate with unsupervised semantic word vectors
TransE(unif) [10]	75.85	None
TransE(bern) [10]	75.82	None
TransE [8]	85.2	Initiate embedding with word2vec
TransH(unif) [10]	77.7	None
TransH (bern) [10]	78.8	None
TranSparse-DT [26]	87.1	None
TransD [9]	86.4	Initiate embedding with the result of TransE
TransR [14]	85.9	Initiate embedding with the result of TransE
CTransR (bern) [14]	85.7	Initiate embedding with the result of TransE
TransG [27]	87.4	Initiate embedding by [28]
ENN	87.3	None
ENN+GNSs	89.4	None
ENN + FCSA	88.2	None
ENN + GNSs + FCSA	87.4	None

Analysis of Generate Positive triples. In this section, the effectiveness of the generated positive examples is analyzed. In this regard, some of the positive samples generated in procedure GNSs are given in Table 3. As shown in table 3, the bold tails are also in the test data set. Adding these positive samples and fine-tuning the triple based neural

network with the new training dataset will increase the accuracy and improve the ranks of the test samples.

Table 3. Samples of Generated New Samples

Generated Positive triples in GNSs
(__chromatic_color_,1 _has_instance, __pink_4)
(__chromatic_color_,1 _has_instance, __red_1)
(__period_1, _has_instance, __bronze_age_1)
(__period_1, _has_instance, __civilisation_2)
(__period_1, _has_instance, __june_1)
(__astronomy_1, _domain_region, __apex_2)
(__astronomy_1, _domain_region, __zenith_1)
(__astronomy_1, _domain_region, __outer_planet_1)
(__family_lobeliaceae_1,_member_meronym, __dicot_family_1)
(__japan_2, _has_part, __hondo_1)

Although triples like *(__period_1,_has_instance,__june_1)* and (__chromatic_color_,1_has_instance, __pink_4) not in the test data set, their tails are in the same community with examples like (__period_1, _has_instance, __season_5) and *(__chromatic_color_1, _has_instance, __yellow_2)* respectively, as a result, according to Figure 6, they have affected the performance of the relationship.

4.5. Results on Wordnet18

This section evaluates and represents results on Wordnet18 in two levels. First, results from the four examinations are presented, then a comprehensive analysis of the results of a variety of evaluation criteria with the other state-of-the-art methods is provided. Table 3 shows the result of four different examinations. In this table, the results are displayed in two raw and filtered modes with evaluation criteria.

MR is quite sensitive to the outliers. From Table 3, we see that different strategies do not have much effect on the outliers and make significant changes. Unlike Wordnet11, applying both of the strategies has decreased the value of MR, which indicates it has advantages in some ways. The lower value of MR, the more desirable. One of the matters is to reduce the rank of the outliers. Although the effect is not striking, cannot ignore. The $Hit@k$ criterion is a significant benchmark, due to it helps to understand the capability of assigning better ranks to each triple. It is essential to be assured, how many potential triples in the K first choices are predicted. Hence, the examinations have been evaluated by $K = 1, 3, 10$ [18]. As illustrated in Table 3, over more than 90% of samples are predicted with $k = 10$. Even in the strictest mode, which $k = 1$, more than half of the samples predict as the first prediction option. An assessment with $k = 3$ is the balance between a flexible and yet rigorous one. However, more than two-thirds of the test cases have been predicted. The combination of ENN and the GNSs strategy has achieved the best value in all evaluation criteria except MR compared to other examinations. It seems that the model has a better performance in increasing the volume of the knowledge base. Although applying the FCSA has a positive effect, does not has a significant performance due to the constant size of the due to the regularization is robust to overfitting and does not need any pre-training and extra optimization functions. It achieves state-of-the-art results on benchmark datasets. Besides, we propose two strategies, GNSs and FCSA, to augment datasets to overcome the heterogeneity of the dataset. In our analysis, we show the performance of applying the knowledge base, which the original triple replaces with the new one. Regarding the application of both strategies on the ENN, the same argument applies to the Wordnet11 dataset. As a conclusion from the experiments in Wordnet18, the number of added triples must be controlled. Obviously, by combining both strategies with the embedded neural network, it cannot allocate very low ranks to triples. On the other hand, it assigns the lower ranks to the outliers [5, 30]. It shows that generated positive triples may be helpful to bring information from other aspects.

In contrast to MR, MRR is insensitive to outliers. The results also show that increasing the size of the knowledge leads to better MRR results. This supports our hypothesis.

Table 4 compares the proposed method with other states of arts. In this table, the types of optimizations used are specified to make better comparisons. The HolE and ComplEx implement each of the comparison methods individually and have performed different optimization functions, which have the results reported for TransE being different from one another and the original article. So, it is difficult to determine precisely how much models with pre-training gain over the other ones [12, 18].

ENN has been able to independently handle the structure of the triple, without any pre-training and additional information to perform better. On the MRR metric, ENN cannot achieve as good performance as the model with pre-training. There are two noticeable phenomena in the result. First, ENN cannot assign a lower rank to the triples. We believe that this phenomenon is caused by the regularization of the models, even though the principle of it has the potential to represent real knowledge and to achieve knowledge graph completion. Second, it shows that an augmented knowledge base affects weaker but consistent improvement on all metrics.

The proposed method has a significant performance compared to non-pre-trained methods, and its results reflect the evaluation criteria of MR, $Hit@10$, and MRR. ENN with GNSs and FCSA largely outperforms on MR and yields a score of 109 among all methods. Since ENN's Regularization cannot assign a lower rank to most of the triples, it can compete with the state-of-the-art model [31, 32].

Table 4. The comparison of results on Wordnet18 with previous work

Methods	Raw					Filtered				
	MR	MRR	Hit@1	Hit@3	hit@10	MR	MRR	Hit@1	hit@3	Hit@10
ENN	120	0.65	37.42	70.9	85.18	115	0.696	46.24	86.64	93.29
ENN+GNSs	116	0.664	42.98	82.02	91.08	113	0.703	50.54	90.1	94.92
ENN + FCSA	117	0.659	39.92	75.22	90.6	111	0.68	46.8	87.2	93.67
ENN + GNSs + FCSA	114	0.643	38.96	73.66	89.21	109	0.679	47.22	86.12	93.34

Table 5. The comparison of results on Wordnet18 with previous work

Methods	Raw		Filtered			Opt
	MR	hit@10	MR	hit@10	mrr	
TansE [13]	263	75.4	251	89.4	-	None
TransE	-	-	-	94.3	0.495	Using Optimize function [12]
TransH [10]	401	73.0	303	86.7	-	None
NTN [16]	-	-	-	66.1	0.53	None
ManifoldE Sphere [29]	-	81.1	-	94.4	-	Initiate embedding by [28]
ManifoldE Hyperplane [29]	-	81.4	-	93.7	-	Initiate embedding by [28]
TransR [14]	238	79.8	225	92.0	-	Initiate embedding with the result of TransE
TransR [14]	-	-	-	94.9	0.605	using optimize function [8]
CTransR (bern) [14]	231	79.4	218	92.3	-	Initiate embedding with the result of TransE
TransD [9]	224	79.6	212	92.2	-	Initiate embedding with the result of TransE
TransG [27]	483	81.4	470	93.3	-	Initiate embedding by [28]
TranSparse-DT [26]	234	81.4	211	94.3	-	None
HolE [18]	-	-	-	94.9	0.938	using optimize function
ComplEx [12]	-	-	-	94.7	0.941	using optimize function
ConvE [31]	-	-	504	94.2	0.955	Use dropout on the embeddings
R-GCN[32]	-	-	-	96.4	0.819	None
TorusE [8]	-	-	-	95.4	0.947	using optimize function
KE-GCN[32]	-	-				
ENN	120	85.18	115	93.29	0.796	None
ENN+GNSs	116	91.08	113	94.92	0.803	None
ENN + FCSA	117	90.6	111	93.67	0.78	None
ENN + GNSs + FCSA	114	89.21	109	93.34	0.679	None

5. Conclusion and future studies

This paper describes a model based on a triple structure for embedding entities and relations via an embedding neural network (ENN). We found that previous methods failed to overfit on infrequent relations. ENN strategies are consistent and reliable. In particular, GNSs and FCSA aren't model dependent, and they can be applied to any models. We believe this observation is essential to assess and prioritize directions for further research on the topic.

In our future work, we will focus on improving the ENN, which needs to utilize loss function. Due to the significant results of the proposed strategies, we will consider other methods for generating new samples and employ them.

6. References

[1] Miller, G. A., "WordNet: a lexical database for English," *Communications of the ACM*, pp. 39-41, 1995.

[2] Suchanek, F. M., Kasneci, G., and Weikum, G., "Yago: a core of semantic knowledge," 2007.

[3] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E., "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, Vol. 104, No. 1, pp. 11-33, 2015.

[4] Sadeghi, A., Graux, D., and Lehmann, J., "MDE: Multi Distance Embeddings for Link Prediction in Knowledge Graphs," *arXiv preprint arXiv:1905.10702*, 2019.

[5] Cai, H., Zheng, V. W., and Chang, K., "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, p. IEEE, 2018.

[6] Ebisu, T., and Ichise, R., "Toruse: Knowledge graph embedding on a lie group," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] Mighan, Sima Naderi, Mohsen Kahani, and Fateme Pourgholamali. "POI Recommendation Based on Heterogeneous Graph Embedding." *2019 9th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE, 2019.

[8] Nguyen, D. Q., Sirts, K., Qu, L., and Johnson, M., "STransE: a novel embedding model of entities and relationships in knowledge bases," in *Proceedings of NAACL-HLT*, 2016.

[9] Ji, G., Liu, K., He, S., and Zhao, J., "Knowledge graph completion with adaptive sparse transfer matrix," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[10] Wang, Z., Zhang, J., Feng, J., and Chen, Z., "Knowledge Graph Embedding by Translating on Hyperplanes," in *AAAI*, 2014.

[11] Li, Zhifei, Hai Liu, Zhaoli Zhang, Tingting Liu, and Neal N. Xiong. "Learning knowledge graph embedding with heterogeneous relation attention networks." *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[12] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G., "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*, 2016.

[13] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O., "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013.

[14] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X., "Learning Entity and Relation Embeddings for Knowledge Graph Completion," in *AAAI*, 2015.

[15] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L., "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[16] Socher, R., Chen, D., Manning, C. D., and Ng, A., "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013.

[17] Abedini, F., Menhaj, M. B., and Keyvanpour, M. R., "Neuron Mathematical Model Representation of Neural Tensor Network for RDF Knowledge Base Completion," *Journal of Computer & Robotics*, Vol. 10, No. 1, pp. 1-10, 2017.

[18] Nickel, M., Rosasco, L., Poggio, T. A., and others, "Holographic Embeddings of Knowledge Graphs," *AAAI*, pp. 1955-1961, 2016.

[19] Haghani, S., and Keyvanpour, M. R., "moLink: Modeling link representation of knowledge base," in *Information and Knowledge Technology (IKT), 2017 9th International Conference on*, 2018.

[20] Cao, X., Wei, Y., Wen, F., and Sun, J., "Face alignment by explicit shape regression," *International Journal of Computer Vision*, Vol. 107, No. 2, pp. 177-190, 2014.

[21] Cui, X., Goel, V., and Kingsbury, B., "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, Vol. 23, No. 9, pp. 1469-1477, 2015.

[22] Bengio, Y., Courville, A., and Vincent, P., "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 35, No. 8, pp. 1798-1828, 2013.

[23] Haghani, S., and Keyvanpour, M. R., "A systemic analysis of link prediction in social network," *Artificial Intelligence Review*, Vol. 52, pp. 1961-1995, 2019.

[24] Keyvanpour, M., Kholghi, M., and Haghani, S., "Hybrid of Active Learning and Dynamic Self-Training for Data Stream Classification," *International Journal of Information & Communication Technology Research*, Vol. 9, No. 4, 2017.

[25] Zhu, J., Jia, Y., Xu, J., and others, "Modeling the Correlations of Relations for Knowledge Graph Embedding," *J. Comput. Sci. & Technol*, Vol. 33, No. 2, 2018.

[26] Chang, L., Zhu, M., Gu, T., Bin, C., Qian, J., and Zhang, J., "Knowledge Graph Embedding by Dynamic Translation," *IEEE Access*, Vol. 5, pp. 20898-20907, 2017.

[27] Xiao, H., Huang, M., and Zhu, X., "TransG: A generative model for knowledge graph embedding," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[28] Glorot, X., and Bengio, Y., "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010.

[29] Xiao, H., Huang, M., and Zhu, X., "From one point to a manifold: Knowledge graph embedding for precise link prediction," *arXiv preprint arXiv:1512.04792*, 2015.

[30] Rosso, Paolo, Dingqi Yang, and Philippe Cudré-Mauroux. "Beyond triplets: hyper-relational knowledge graph embedding for link prediction." *Proceedings of The Web Conference* 2020.

[31] Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S., "Convolutional 2d knowledge graph embeddings," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[32] Schlichtkrull, M. a. K. T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M., "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, 2018.

[33] Yu, Donghan, et al. "Knowledge embedding based graph convolutional network." *Proceedings of the Web Conference 2021*. 2021

[34] Kazemi, Seyed Mehran, and David Poole. "Simple embedding for link prediction in knowledge graphs." *Advances in neural information processing systems* 31, 2018.

[35] Sun, Zhiqing, et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." *arXiv preprint arXiv:1902.10197*, 2019.

[36] Nguyen, Dat Quoc. "An overview of embedding models of entities and relationships for knowledge base completion." *arXiv preprint arXiv:1703.08098*, 2017