Efficient Non-Sampling Knowledge Graph Embedding

Zelong Li*
Rutgers University
New Brunswick, NJ, US
zelong.li@rutgers.edu

Jianchao Ji*
Rutgers University
New Brunswick, NJ, US
jianchao.ji@rutgers.edu

Zuohui Fu
Rutgers University
New Brunswick, NJ, US
zuohui.fu@rutgers.edu

Yingqiang Ge
Rutgers University
New Brunswick, NJ, US
yingqiang.ge@rutgers.edu

Shuyuan Xu
Rutgers University
New Brunswick, NJ, US
shuyuan.xu@rutgers.edu

Chong Chen
Tsinghua University
Beijing, China
cc17@mails.tsinghua.edu.cn

Yongfeng Zhang
Rutgers University
New Brunswick, NJ, US
yongfeng.zhang@rutgers.edu

ABSTRACT

Knowledge Graph (KG) is a flexible structure that is able to describe the complex relationship between data entities. Currently, most KG embedding models are trained based on negative sampling, i.e., the model aims to maximize some similarity of the connected entities in the KG, while minimizing the similarity of the sampled disconnected entities. Negative sampling helps to reduce the time complexity of model learning by only considering a subset of negative instances, which may fail to deliver stable model performance due to the uncertainty in the sampling procedure. To avoid such deficiency, we propose a new framework for KG embedding—Efficient Non-Sampling Knowledge Graph Embedding (NS-KGE). The basic idea is to consider all of the negative instances in the KG for model learning, and thus to avoid negative sampling. The framework can be applied to square-loss based knowledge graph embedding models or models whose loss can be converted to a square loss. A natural side-effect of this non-sampling strategy is the increased computational complexity of model learning. To solve the problem, we leverage mathematical derivations to reduce the complexity of non-sampling loss function, which eventually provides us both better efficiency and better accuracy in KG embedding compared with existing models. Experiments on benchmark datasets show that our NS-KGE framework can achieve a better performance on efficiency and accuracy over traditional negative sampling based models, and that the framework is applicable to a large class of knowledge graph embedding models.

CCS CONCEPTS

• **Computing methodologies → Knowledge representation and reasoning**; **Learning latent representations**; *Neural networks.*

KEYWORDS

Knowledge Graph Embedding; Non-Sampling Machine Learning; Computational Efficiency; Space Efficiency

[*] The first two authors contributed equally to the work.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449859

ACM Reference Format:
Zelong Li, Jianchao Ji, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Chong Chen, Yongfeng Zhang. 2021. Efficient Non-Sampling Knowledge Graph Embedding. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449859

1 INTRODUCTION

Nowadays, Knowledge Graph (KG) is an important structure to store and process structured information, and has been widely used as an advanced knowledge interlinking solution in many applications. Concretely, KG is a collection of interlinked descriptions of entities. For example, Freebase, which is regarded as a practical and massive tuple database used to structure general human knowledge, is powered by KG algorithms [4]. Yet another Great Ontology (YAGO) also benefits from KG structures when building its light-weight and extensible ontology with high coverage and quality [35]. Moreover, DBpedia builds a large-scale, multilingual knowledge base by extracting structured data from Wikipedia editions in 111 languages [26]. These knowledge bases power a wide range of intelligent systems in practice.

Although KG has been proved as an effective method to represent large-scale heterogeneous data [23], it suffers from high computation and space cost when searching and matching the entities in a discrete symbolic space. In order to leverage the power of KG more efficiently, researchers have proposed Knowledge Graph Embedding (KGE), which represents and manipulates KG entities in a latent space [6]. In particular, KGE techniques embed the components of KG, including entities and relations, into a continuous vector space, so as to simplify the manipulation while preserving the inherent structure of the KG [41]. With the help of KGE, implementing knowledge graph operations to a large scale becomes practical.

Over the past few years, a lot of efforts have been put into developing embedding algorithms, and a growing number of embedding models are proven effective. Most of the current embedding methods, however, depend on a very basic operation in model training called negative sampling [6, 34], which randomly or purposely samples some disconnected entities as negative samples (compared to the connected entities as positive samples), and the embedding model aims to distinguish positive vs. negative samples in the loss function for embedding learning. Typical examples include DistMult [48], SimplE [24], ComplEx [37], TransE [6], RESCAL [31], etc. While negative sampling increases the training efficiency, it also brings several drawbacks in model accuracy. On one hand, only considering part of the negative instances weakens the prediction accuracy of the learned embeddings. And on the other hand, it also makes model training unstable because the sampled negative instances may vary in different runs. Previous studies have shown that negative sampling keeps bringing fluctuations since the result highly relies on the selection of negative samples [38], and these fluctuations cannot be removed by doing more updating steps [10]. Some research tried to overcome this problem by using carefully designed sampling strategies instead of random sampling [7, 52]. However, all of these models can only consider part of the information from the training dataset due to negative sampling.

Inspired by recent progress on non-sampling recommendation and factorization machines [8, 9, 11–13, 32, 33], we make an attempt to apply the non-sampling approach to KGE. We propose a Non-Sampling Knowledge Graph Embedding (NS-KGE) framework. The framework can be applied to square loss based KGE models. To apply this framework to models with other loss functions, we need to transform the loss function to square loss. The framework allows us to skip the negative sample selection process and thus take all of the positive and negative instances into consideration when learning the knowledge graph embeddings, which helps to increase the embedding accuracy. A problem that naturally arises from this strategy is the time and space complexity increase dramatically when considering all instances for embedding learning. To solve the problem, we offer a mathematical derivation to re-write the non-sampling loss function by dis-entangling the interaction between entities, which gives us better time and space complexity without sacrificing of mathematical accuracy. Eventually, non-sampling based KGE achieves better prediction accuracy with similar space and much shorter running time than existing negative sampling based KGE models. To evaluate the performance of our NS-KGE framework, we apply the framework on four KGE models, including DistMult [48], SimplE [24], ComplEx [37], and TransE [6]. Experimental results show that the NS-KGE framework outperforms most of the models in terms of both prediction accuracy and learning efficiency.

This paper makes the following key contributions:

- We propose a Non-Sampling Knowledge Graph Embedding (NS-KGE) framework for learning effective knowledge graph embeddings.
- We derive an efficient method to mitigate the time and space bottlenecks caused by the non-sampling strategy.
- We demonstrate how the framework can be mathematically applied to existing KGE models by using DistMult [48], SimplE [24], ComplEx [37], and TransE [6] as examples.
- We conduct comprehensive experiments—including both quantitative and qualitative analyses—to show that the framework increases both accuracy and efficiency for knowledge graph embedding.

In the following part of this paper, we first introduce the related work in Section 2. In Section 3, we introduce our NS-KGE framework in detail, and in Section 4 we show how the framework can be applied to different KGE models. We provide and analyze the experimental results in Section 5, and conclude the work together with future directions in Section 6.

2 RELATED WORK

During recent years, Knowledge Graph Embedding (KGE) has prevailed in the field of huge structured knowledge interlinking [42], and its effectiveness has been shown in many different scenarios such as search engine [2, 15, 46], recommendation system [1, 16, 29, 39, 44, 45, 49], question answering [22, 27, 47], video understanding [18], conversational AI [17, 53] and explainable AI [1, 2, 51].

Since knowledge graph embedding has extraordinary advantages in practical applications, many KGE models have been proposed. For example, Translational Embedding (TransE) model [6] takes vector translation on spheres to model entity relationships, Translation on Hyperplane (TransH) model [43] enables vector translation on hyperplanes for embedding, while Translation in Relation spaces (TransR) model [28] conducts vector translation in relation-specific entity spaces for embedding. Later, DistMult [48] uses a diagonal matrix to represent the relation between head and tail entities, and the composition of relations is characterized by matrix multiplication, while ComplEx [37] puts DistMult into the complex domain and uses complex numbers to represent the head-relation-tail triples in the knowledge graph. More recently, SimplE [24] provides a simple enhancement of the Canonical Polyadic (CP) [20] tensor factorization for interpretable knowledge graph embedding. More comprehensive review of knowledge graph embedding techniques can be seen in [21, 42].

Most of the existing KGE models rely on negative sampling for model learning, which randomly sample some disconnected entities to distinguish with connected entities, and meanwhile reduce the training time compared with using all negative samples. However, due to the uncertainty of sampling negative instances, the results of embedding learning may fluctuate greatly in different runs. Besides, some models only produce satisfying embedding results when the number of negative samples is large enough [37, 48], which increases the time needed for model training.

In previous works, some methods [40, 50] have been developed to mitigate the above problems, mostly by sampling the negative instances purposely rather than randomly. For example, dynamic negative sampling [50] chooses negative training instances from the ranking list produced by the current prediction model, so that the model can continuously work on the most difficult negative instances. Generative Adversarial Networks (GAN) are also used to generate and discriminate high-quality negative samples [40], which take advantage of a generator to obtain high-quality negative samples, and meanwhile the discriminator in GAN learns the embeddings of the entities and relations in knowledge graph so as to incorporate the GAN-based framework into various knowledge grahp embedding models for better ability of knowledge representation learning. Ultimately, these models still rely on the sampled negative instances instead of all instances for model training and the model accuracy still has room for improvement.

Recently, researchers have explored whole-data based approaches to train recommendation models [8, 11, 12], which improve the recommendation accuracy without negative sampling. By separating the users and items in optimization, the computational bottlenecks has been resolved in a large extent in training the recommendation models. However, these methods can only be applied to recommendation models, while we would like to build a general framework that can be applied to square-loss-based knowledge graph embedding models. Besides, although these models achieve better recommendation performance and efficiency, they do not consider improving the space complexity but only focus on the time complexity, and thus they still need to use batch learning during the training process. On the contrary, we aim to improve both the space and time complexity in this work. Based on this, we can achieve three benefits: better entity ranking accuracy, better computational efficiency, and better space efficiency.

3 NON-SAMPLING KGE FRAMEWORK

In this section, we will first introduce the notations that will be used in this paper. Then, we will introduce the Non-Sampling Knowledge Graph Embedding (NS-KGE) framework step by step. In particular, we will first provide a general formulation of the framework, and then devote two subsections to show how to improve the time and space efficiency in the framework. We will show how the framework can be applied to different specific knowledge graph embedding models in the next section.

3.1 Problem Formalization and Notations

In this section, we provide a square-loss based formalization of the Knowledge Graph Embedding (KGE) problem, which will be used in the following parts of the paper. However, we acknowledge that not all of the existing KGE methods can be represented by this square loss formalization. In this paper, we consider those KGE methods whose loss function is a square loss or can be converted into a square loss format for non-sampling KGE learning. Table 1 introduces the basic notations that will be used in this paper. We first provide a general formulation for the knowledge graph embedding problem. Given a knowledge graph $\boldsymbol{G}$, our goal is to train a scoring function $\hat{f}_r(h,t)$, which is able to distinguish whether the head entity h and tail entity t should be connected by relation r in the knowledge graph. Suppose $f_r(h,t)$ is the ground-truth value of the triplet (h,r,t), generated from training sets, and $\hat{f}_r(h,t)$ is the predicted value by the knowledge graph embedding model, where $f_r(h,t) = 1$ represents the connected entities, and $f_r(h,t) = 0$ denotes dis-connected entities. Based on these definition, a general KG embedding model aims to minimize the difference between the ground-truth and the predicted values based on a loss function L. For example, we can use square loss to train the model:

$$L = \sum_{r\in \boldsymbol{R}} \sum_{h\in \boldsymbol{E}} \sum_{t\in \boldsymbol{E}} c_{hrt} \left(f_r(h,t) - \hat{f}_r(h,t) \right)^2 \tag{1}$$

where the three summations enumerate all of the possible (h,r,t) triplet combinations in the knowledge graph, and c_{hrt} represents the importance (i.e., weight score) of the corresponding triplet. In traditional negative sampling-based KGE models such as TransE, $c_{hrt} = 1$ is set as the positive instances and the sampled negative instances, while for all other negative instances, $c_{hrt} = 0$. In our non-sampling KGE framework, however, all c_{hrt} values are non-zero. In the simplest case, $c_{hrt} = 1$ for all instances, regardless of positive or negative.

Symbol	Description
$\boldsymbol{G}$	A knowledge graph
$\boldsymbol{E}$	The set of entities in a knowledge graph
$\boldsymbol{R}$	The set of relations in a knowledge graph
h, t	A head (h) or a tail (t) entity in a knowledge graph
r	A relation in a knowledge graph
$\boldsymbol{e}_h, \boldsymbol{e}_t$	Embedding vector of the entity h and t
$\boldsymbol{r}$	Embedding vector of the relation r
$e_{h,i}, e_{t,i}$	The i-th dimension of entity embedding $\boldsymbol{e}_h$ and $\boldsymbol{e}_t$
r_i	The i-th dimension of relation embedding $\boldsymbol{r}$
d	Dimension of the embedding vectors
c_{hrt}	The weight of the triplet (h,r,t)
$f_r(h,t)$	Ground-truth value of the triple (h,r,t)
$\hat{f}_r(h,t)$	Predicted value of the triple (h,r,t)

Table 1: Summary of the notations in this work.

Many knowledge graph embedding models can be regarded as a special case of this formulation. For example, TransE uses $\hat{f}_r(h,t) = \|\boldsymbol{e}_h + \boldsymbol{r} - \boldsymbol{e}_t\|$ as the scoring function. For connected entities, the ground-truth value $f_r(h,t)$ would be 0, and for dis-connected entities, the ground-truth value would be a constant value greater than 0 (e.g., it would be 3 when $\boldsymbol{e}_h, \boldsymbol{r}$ and $\boldsymbol{e}_t$ are regularized as unit vectors). Similar to TransE, many other KG embedding models can be represented by Eq.(1) with no or only a little trivial transformation, as we will show later in Section 4.

3.2 Non-sampling KG Embedding

The adoption of square loss in Eq.(1) makes it possible to simplify the model learning and increase the time and space efficiency based on mathematical re-organization of the loss function. In the first step, we can re-write the loss function as following:

$$\begin{aligned} L &= \sum_{r\in \boldsymbol{R}} \sum_{h\in \boldsymbol{E}} \sum_{t\in \boldsymbol{E}} c_{hrt} \left(f_r(h,\, t) - \hat{f}_r(h,\, t) \right)^2 \\ &= \sum_{r\in \boldsymbol{R}} \sum_{h\in \boldsymbol{E}} \sum_{t\in \boldsymbol{E}} c_{hrt} \left(f_r(h,\, t)^2 + \hat{f}_r(h,\, t)^2 - 2 f_r(h,\, t)\hat{f}_r(h,\, t) \right) \end{aligned} \tag{2}$$

From Eq.(2), we can see that the time complexity of calculating the loss is huge. The time complexity of calculating $\hat{f}_r(h,t)$ is $O(d)$, where d is the dimension of embedding vectors, and thus the time complexity of calculating the whole loss function is $O(d|\boldsymbol{R}||\boldsymbol{E}|^2)$. If we implement this loss function on real-world knowledge graphs, we would have to conduct trillions of times of computation to calculate the loss function within one epoch. Depending on the size of the training data, this may take days, weeks or even longer to train a model. Even if using all of the (both positive and negative) samples in the dataset for model training can bring us better accuracy, such training time is not affordable. As a result, we need to mathematically derive more efficient implementations for Eq. (2).

3.3 Improving Time Efficiency

To reduce the time complexity, the first thing we need to consider is to find out the most time-consuming part of the loss function. Without loss of generality and to simplify the model computation, we assume ground-truth value $f_r(h,t) = 1$ for positive instances and $f_r(h,t) = 0$ for negative instances. Besides, we set a uniform coefficient c^+ for all positive instances and c^- for all negative instances.

In this case the loss function can be re-organized as:

$$
\begin{aligned}
L &= \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}} c_{hrt}\big(f_r(h,t)-\hat{f}_r(h,t)\big)^2 \\
&\overset{1}{=} \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\Big[\sum_{t\in\boldsymbol{E}^+_{h,r}} c^+\Big(f_r(h,t)^2+\hat{f}_r(h,t)^2-2f_r(h,t)\hat{f}_r(h,t)\Big) \\
&\quad + \sum_{t\in\boldsymbol{E}^-_{h,r}} c^-\hat{f}_r(h,t)^2\Big] \\
&\overset{2}{=} \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\Big[\sum_{t\in\boldsymbol{E}^+_{h,r}} c^+\Big(f_r(h,t)^2+\hat{f}_r(h,t)^2-2f_r(h,t)\hat{f}_r(h,t)\Big) \\
&\quad + \Big(\sum_{t\in\boldsymbol{E}} c^-\hat{f}_r(h,t)^2-\sum_{t\in\boldsymbol{E}^+_{h,r}} c^-\hat{f}_r(h,t)^2\Big)\Big] \\
&\overset{3}{=} \underbrace{\sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}^+_{h,r}}\Big[c^+\Big(\hat{f}_r(h,t)^2-2f_r(h,t)\hat{f}_r(h,t)\Big)-c^-\hat{f}_r(h,t)^2\Big]}_{L^P} \\
&\quad + \underbrace{\sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}} c^-\hat{f}_r(h,t)^2}_{L^A} + \underbrace{\sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}^+_{h,r}} c^+f_r(h,t)^2}_{\text{constant}}
\end{aligned}
\tag{3}
$$

where $\boldsymbol{E}^+_{h,r}$ represents the set of entities in the KG that are connected to head entity h by relation r, while $\boldsymbol{E}^-_{h,r}$ is the set of entities that are not connected to h through r. We have $\boldsymbol{E}^+_{h,r} \cup \boldsymbol{E}^-_{h,r} = \boldsymbol{E}$. In the step 1 above, we split the loss function by considering $\boldsymbol{E}^+_{h,r}$ and $\boldsymbol{E}^-_{h,r}$ separately. In step 2, we replace the $\boldsymbol{E}^-_{h,r}$ term by subtracting $\boldsymbol{E}^+_{h,r}$ from the total summation, and in step 3, we re-organize the loss function into the positive term L^P, all entity term L^A, and a constant value term. In the loss function, we separate the positive entities and the constant value, and we use all of the data to replace the negative samples. In the next step, we will introduce how to optimize the time complexity after such transformation.

As we can see, the L^P term enumerates over all of the connected triplets in the KG, and its time complexity is $O(d\times\#$ positive triples$)$, which is an acceptable complexity. However, since most of the KG datasets are highly sparse, L^A, even with pretty concise form, contributes the most significant time complexity to the loss function. Actually, the time complexity of L^A is $O(d|\boldsymbol{R}||\boldsymbol{E}|^2)$, which is very expensive. We hope the time complexity of L^A can be further reduced. As a result, we will take a closer look at the $\hat{f}_r(h,t)^2$ term.

Fortunately, NS-KGE can be applied to most of the factorization-based KG embedding models. For these models, we can conduct certain transformations over the scoring function $\hat{f}_r(h,t)$ to reduce the time complexity. For a factorization-based KG embedding model, the scoring function $\hat{f}_r(h,t)$ can be represented as the following general formulation:

$$
\hat{f}_r(h,t) = \boldsymbol{e}_h^T(\boldsymbol{r}\odot\boldsymbol{e}_t) = \sum_i^d e_{h,i}r_ie_{t,i}
\tag{4}
$$

where $\boldsymbol{e}_h$, $\boldsymbol{r}$ and $\boldsymbol{e}_t$ are the head entity embedding, relation embedding, and tail entity embedding, respectively, and the symbol $\odot$ denotes element-wise product. Besides, $e_{h,i}$, r_i and $e_{t,i}$ denote the i-th element of the corresponding embedding vector. Since our NS-KGE framework is based on square loss, we calculate the square of $\hat{f}_r(h,t)$. By manipulating the inner-product operation, the square of $\hat{f}_r(h,t)$ can be rearranged as:

$$
\begin{aligned}
\hat{f}_r(h,t)^2 &= \Big(\sum_i^d e_{h,i}r_ie_{t,i}\Big)\Big(\sum_j^d e_{h,j}r_je_{t,j}\Big) \\
&= \Big(\sum_i^d\sum_j^d e_{h,i}e_{h,j}\Big)\Big(\sum_i^d\sum_j^d r_ir_j\Big)\Big(\sum_i^d\sum_j^d e_{t,i}e_{t,j}\Big)
\end{aligned}
\tag{5}
$$

In this case, $\boldsymbol{e}_h$, $\boldsymbol{r}$ and $\boldsymbol{e}_t$ are separated from each other, and thus $\sum_i^d\sum_j^d e_{h,i}e_{h,j}$, $\sum_i^d\sum_j^d r_ir_j$ and $\sum_i^d\sum_j^d e_{t,i}e_{t,j}$ are independent from each other. Therefore, we can disentangle the parameters and calculate $\hat{f}_r(h,t)^2$ in a more efficient way. We thus apply Eq.(5) to the L^A term of the loss function Eq.(3) and we have:

$$
\begin{aligned}
\boldsymbol{L} &= L^P + c^-\sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}}\Big(\sum_i^d\sum_j^d e_{h,i}e_{h,j}\Big)\Big(\sum_i^d\sum_j^d r_ir_j\Big)\Big(\sum_i^d\sum_j^d e_{t,i}e_{t,j}\Big) \\
&= L^P + c^-\overbrace{\sum_i^d\sum_j^d\underbrace{\Big(\sum_{h\in\boldsymbol{E}} e_{h,i}e_{h,j}\Big)}_{L_H}\underbrace{\Big(\sum_{r\in\boldsymbol{R}} r_ir_j\Big)}_{L_R}\underbrace{\Big(\sum_{t\in\boldsymbol{E}} e_{t,i}e_{t,j}\Big)}_{L_T}}^{L^A}
\end{aligned}
\tag{6}
$$

In the second step of Eq.(6), the reason that we can reorganize the summary of i, j and the summary over h, r, t is because the L_H, L_R and L_T terms are independent from each other. We also leave out the constant term since it does not influence the optimization result.

As noted before, L^A contributes the most significant complexity to the loss function. Based on the above operation, the complexity of L^A is reduced from $O(d|\boldsymbol{E}||\boldsymbol{R}||\boldsymbol{E}|)$ in Eq.(3) to $O(d^2(|\boldsymbol{E}|+|\boldsymbol{R}|+|\boldsymbol{E}|))$ in Eq.(6). In Section 4, we take the Bilinear-Diagonal embedding model (DistMult), the Simple enhancement of Canonical Polyadic model (SimplE), the Complex Embedding model (ComplEx) and the Translational Embedding model (TransE) as examples to show how the NS-KGE framework can be applied to different models.

3.4 Improving Space Efficiency

Apart from time complexity, we also provide a method to reduce the space complexity which is still based on the factorization-based scoring function $\hat{f}_r(h,t)$ in Eq.(4). The models that will be studied in Section 4 satisfy this form or could be extended with some simple transformations.

First, we use two $|E|\times d$ matrices, H_e and T_e, to store the embedding vectors of all head entities and tail entities, respectively. Similarly, we store the embedding vectors of all relations in the matrix R_e, with the size of $|R|\times d$. According to Eq.(6), the calculation of L_H, L_R and L_T are independent from each other with the calculation of each term only relies on the corresponding H_e, R_e or T_e matrix. For example, given the index i and j, the value of the L_H term is equal to the inner product of the i-th column vector and the j-th column vector of matrix H_e. As a result, we can construct three intermediate matrices denoted as M_H, M_T, and M_R with the size $d\times d$ to record the intermediate results for each term. The details are displayed in Eq.(7).

$$
M_H = H_e^TH_e,\ M_R = R_e^TR_e,\ M_T = T_e^TT_e
\tag{7}
$$

Note that $M_H[i][j]$ is equal to the inner product of the i-th and the j-th column vector of H_e, similar for $M_R[i][j]$ and $M_T[i][j]$. Based on this, the calculation of the L^A term in Eq.(6) can be simplified as:

$$L^A = c^- sum(M_H \odot M_R \odot M_T) \tag{8}$$

where $\odot$ means element-wise product of matrices, and *sum* means adding up all elements of a matrix. In this way, we can calculate L^A in the space complexity of $O\left(d \times (|R| + |E| + d)\right)$, so that we do not need to use any batch optimization for standard knowledge graph benchmarks such as the FB15K237 and WN18RR datasets (to be introduced in Section 5), since we can directly use the whole training data to calculate the loss function within reasonable time and space complexity.

As will be shown in the following section, for different models we may need to construct the M_H, M_R and M_T matrices in different ways, but this does not increase the space complexity. Besides, for extremely large datasets that cannot be loaded into memory as a whole, our framework with smaller space complexity can use fewer batches to train the model, which results in less training epochs.

4 APPLY NS-KGE ON DIFFERENT MODELS

In this section, we will show how our NS-KGE framework can be applied over different KGE models. As mentioned before, we will take the Bilinear-Diagonal embedding model (DistMult) [48], the Simple enhancement of Canonical Polyadic model (SimplE) [24], the Complex Embedding model (ComplEx) [37] and the Translational Embedding model (TransE) [6] as examples.

4.1 Bilinear-Diagonal Embedding (DistMult)

DistMult is a representative factorization-based multi-relation representation learning model [48]. It learns each entity as a vector embedding, and learns each relation as a diagonal matrix. For a triplet (h, r, t), DistMult trains the model based on the following scoring function:

$$\hat{f}_r(h,t) = \boldsymbol{e}_h^T \cdot diag(\boldsymbol{r}) \cdot \boldsymbol{e}_t = \sum_i^d e_{h,i} r_i e_{t,i} \tag{9}$$

We can see that the scoring function of DistMult is the same as our framework (Eq.(4)). As a result, we can directly apply the loss function of the NS-KGE framework (Eq.(6)) for model learning, and use Eq.(8) for better space complexity. The final time complexity is $O\big(d^2(|\boldsymbol{E}| + |\boldsymbol{R}| + |\boldsymbol{E}|)\big)$. In the experiments, we will show that our Non-sampling DistMult is both more efficient and more effective that the original sampling-based DistMult model.

4.2 Simple Enhancement of CP (SimplE)

Canonical Polyadic (CP) decomposition is one of the earliest work on tensor factorization approaches [20]. Since CP learns two independent embedding vectors for each entity, it performs poorly in KG link prediction tasks. The SimplE [24] embedding method provides a simple improvement of CP by learning the two embeddings of each entity dependently, which gains much better performance on link prediction.

For SimplE, its scoring function is a little bit more complex than DistMult. The scoring function is a combination of two parts:

$$\begin{aligned}\hat{f}_r(h,t) &= \frac{1}{2}\Big(\boldsymbol{e}_h^T(\boldsymbol{r} \odot \boldsymbol{e}_t) + \boldsymbol{e}_t^T(\boldsymbol{r}^{-1} \odot \boldsymbol{e}_h)\Big) \\ &= \frac{1}{2}\left(\sum_{i=1}^d e_{h,i} r_i e_{t,i} + \sum_{j=1}^d e_{t,j} r_j^{-1} e_{h,j}\right)\end{aligned} \tag{10}$$

To apply the NS-KGE framework on SimplE, we need to consider $\hat{f}_r(h,t)^2$, which consists of three terms:

$$\begin{aligned}\hat{f}_r(h,t)^2 = \frac{1}{4}\Bigg(&\sum_i^d \sum_j^d (e_{h,i}e_{h,j})(r_i r_j)(e_{t,i}e_{t,j}) \\ &+2\sum_i^d \sum_j^d (e_{h,i}e_{h,j})(r_i r_j^{-1})(e_{t,i}e_{t,j}) \\ &+\sum_i^d \sum_j^d (e_{h,i}e_{h,j})(r_i^{-1} r_j^{-1})(e_{t,i}e_{t,j})\Bigg)\end{aligned} \tag{11}$$

We can see that for each term in Eq.(11), its structure is the same as that in Eq.(5). As a result, the rest of the work is similar to what we did in Eq.(6). The only difference is that we result in three L^A terms in Eq.(6), but the time and space complexity remain unchanged.

4.3 Complex Embeddings (ComplEx)

The ComplEx embedding model learns KG embeddings in a complex number space [37]. It adopts the Hermitian dot product to construct the scoring function. But we can still do similar rearrangements for non-sampling knowledge graph embedding. The scoring function of ComplEx is:

$$\begin{aligned}\hat{f}_r(h,t) = \boldsymbol{h}_{re}^T(\boldsymbol{r}_{re} \odot \boldsymbol{t}_{re}) + \boldsymbol{h}_{im}^T(\boldsymbol{r}_{re} \odot \boldsymbol{t}_{im}) \\ + \boldsymbol{h}_{re}^T(\boldsymbol{r}_{im} \odot \boldsymbol{t}_{im}) - \boldsymbol{h}_{im}^T(\boldsymbol{r}_{im} \odot \boldsymbol{t}_{re})\end{aligned} \tag{12}$$

where $\boldsymbol{h}_{re}, \boldsymbol{r}_{re}, \boldsymbol{t}_{re}$ are the real part of the head, relation, and tail embedding vectors, while $\boldsymbol{h}_{im}, \boldsymbol{r}_{im}, \boldsymbol{t}_{im}$ are the imaginary part of the head, relation, and tail embedding vectors. There will be six terms for $\hat{f}_r(h,t)^2$. In the following, we would only show the expansion result of the first term $L_1^A = \boldsymbol{h}_{re}^T(\boldsymbol{r}_{re} \odot \boldsymbol{t}_{re}) \cdot \boldsymbol{h}_{im}^T(\boldsymbol{r}_{re} \odot \boldsymbol{t}_{im})$, since the other terms look similar.

$$\begin{aligned}L_1^A &= c^- \sum_{r \in \boldsymbol{R}} \sum_{h \in \boldsymbol{E}} \sum_{t \in \boldsymbol{E}} \Big(\sum_i^d h_{re,i} r_{re,i} t_{re,i}\Big)\Big(\sum_j^d h_{im,j} r_{re,j} t_{im,j}\Big) \\ &= c^- \sum_i^d \sum_j^d \Big(\sum_{h \in \boldsymbol{E}} h_{re,i} h_{im,j}\Big)\Big(\sum_{r \in \boldsymbol{R}} r_{re,i},\, r_{re,j}\Big)\Big(\sum_{t \in \boldsymbol{E}} t_{re,i},\, t_{im,j}\Big)\end{aligned} \tag{13}$$

In this way, we can calculate the loss of non-sampling ComplEx within $O(d^2(|R|+|E|))$ time complexity and $O(d(|R|+|E|+d))$ space complexity.

4.4 Translational Embedding (TransE)

Unlike the previous factorization-based knowledge graph embedding models, TransE [6] is a translation-based embedding model. As a result, the NS-KGE framework cannot be directly applied to

TransE. However, we will show that by applying very minor modifications to the scoring function without changing the fundamental idea of translational embedding, the NS-KGE framework can still be applied to TransE.

The original scoring function of TransE is $\hat{f}_r(h,t) = \|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\|$, which calculates the distance between $\boldsymbol{h}+\boldsymbol{r}$ and $\boldsymbol{t}$. For positive examples, we hope the distance would be as small as possible, which could be 0 in the most ideal case. For negative examples, we hope the distance would be as far away as possible. In many implementations of TransE, to avoid over-fitting, we usually apply unit-vector constraints to the embedding vectors, i.e., $\boldsymbol{h}, \boldsymbol{r}$ and $\boldsymbol{t}$ are regularized as length-one vectors. In this case, the maximum possible distance between $\boldsymbol{h}+\boldsymbol{r}$ and $\boldsymbol{t}$ would be 3, as a result, the optimal value for negative examples would be 3. However, the mathematical derivation of our framework in Eq.(3) relies on the assumption that the ground-truth value for positive instances is 1 and for negative instances it is 0. As a result, we slightly modify the scoring function of TransE to a new function $\hat{f}'_r(h,t) = 1-\frac{1}{3}\hat{f}_r(h,t) = 1-\frac{1}{3}\|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\|$. In this way, the ground-truth value satisfy our assumption meanwhile it does not influence the optimization of TransE. The only difference is that instead of minimizing $\hat{f}_r(h,t)$ for positive examples in TransE, we aim to maximize $\hat{f}'_r(h,t)$, which is basically the same in terms of optimization.

By inserting $\hat{f}'_r(h,t)$ into Eq.(3), we have:

$$
\begin{aligned}
L &= L^P + \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}} c^-\Big(1-\frac{1}{3}\hat{f}_r(h,t)\Big)^2 \\
&= L^P + \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}} c^-\left(\frac{2}{3}\Big(\boldsymbol{h}^T\boldsymbol{t}+\boldsymbol{r}^T(\boldsymbol{t}-\boldsymbol{h})\Big)\right)^2 \\
&= L^P + \sum_{r\in\boldsymbol{R}}\sum_{h\in\boldsymbol{E}}\sum_{t\in\boldsymbol{E}} \frac{4c^-}{9}\Big((\boldsymbol{h}^T\boldsymbol{t})^2+(\boldsymbol{r}^T\boldsymbol{t})^2+(\boldsymbol{r}^T\boldsymbol{h})^2-2\boldsymbol{r}^T\boldsymbol{t}\boldsymbol{r}^T\boldsymbol{h}\Big) \\
&= L^P + \frac{4c^-}{9}\Big(\sum_i^d\sum_j^d |E|\sum_{r\in\boldsymbol{R}} r_ir_j\sum_{t\in\boldsymbol{E}} t_it_j + \sum_i^d\sum_j^d |E|\sum_{h\in\boldsymbol{E}} h_ih_j\sum_{r\in\boldsymbol{R}} r_ir_j \\
&+ \sum_i^d\sum_j^d |R|\sum_{h\in\boldsymbol{E}} h_ih_j\sum_{t\in\boldsymbol{E}} t_it_j - 2\sum_i^d\sum_j^d\sum_{r\in\boldsymbol{R}} r_ir_j\sum_{h\in\boldsymbol{E}} h_i\sum_{t\in\boldsymbol{E}} t_j\Big) \\
&= L^P + \frac{4c^-}{9}\sum_i^d\sum_j^d\Big(|E|\underbrace{\sum_{r\in\boldsymbol{R}} r_ir_j}_{L_R}\underbrace{\sum_{t\in\boldsymbol{E}} t_it_j}_{L_T} + |E|\underbrace{\sum_{h\in\boldsymbol{E}} h_ih_j}_{L_H}\underbrace{\sum_{r\in\boldsymbol{R}} r_ir_j}_{L_R} \\
&+ |R|\underbrace{\sum_{h\in\boldsymbol{E}} h_ih_j}_{L_H}\underbrace{\sum_{t\in\boldsymbol{E}} t_it_j}_{L_T} - 2\underbrace{\sum_{r\in\boldsymbol{R}} r_ir_j}_{L_R}\underbrace{\sum_{h\in\boldsymbol{E}} h_i}_{S_H}\underbrace{\sum_{t\in\boldsymbol{E}} t_j}_{S_T}\Big)
\end{aligned}
\tag{14}
$$

We can see that similar to Eq.(6), the final loss can also be decomposed to the L_H, L_R and L_T terms, which are independent from each other for better time complexity, and can be calculated with better space complexity (Section 3.4). The S_H and S_T terms are just summation of the entity embedding matrix, whose calculation is even easier than the L_H and L_T terms. The final time complexity of non-sampling TransE is $O\left(d^2(|R|+|E|)\right)$.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate both the efficiency and effectiveness of the NS-KGE framework.[1]

Dataset	#entities	#relations	#train	#test
FB15K237	14,541	237	272,115	20,466
WN18RR	40,943	11	86,835	3,134

Table 2: Basic statistics of the datasets

5.1 Experimental Setup

5.1.1 ***Dataset Description.*** We conduct the experiments on two benchmark datasets for knowledge graph embedding research, namely, FB15K237 and WN18RR. The detailed statistics of the datasets are shown in Table 2, and we will briefly introduce these two datasets in the following.

FB15K237: One of the most frequently used dataset for KGE. The original version of the FB15K dataset is generated from a subset of the Freebase knowledge graph [5]. However, in the original FB15K dataset, a large number of the test triplets can be obtained by simply reversing the triplets in the training set, as shown in [3, 36]. For example, the test set may contain a triplet (home, car, work), while the training set contains a reverse triplet (work, car, home). The existence of such cases make the original dataset suffer from the test leakage problem. As a result, the FB15k237 [36] dataset is introduced by removing these reverse triplets, which mitigates the test leakage problem in a large extend. In this paper we use FB15k237 for evaluation.

WN18RR: WN18 is also a standard dataset for KGE. The original WN18 dataset is a subset of WordNet [30], which is an English lexical database. Similar to FB15K, WN18 is also corrected to WN18RR [14] by removing the reverse triplets to avoid test leakage. In this work, we use WN18RR for evaluation.

We use the default train-test split of the original datasets. Both the training set and the testing set are a set of (h, r, t) triplets. The number of training and testing triplets of the two datasets are shown in Table 2.

5.1.2 ***Baselines.*** We study the performance of the NS-KGE framework by comparing the performance of a KGE model with or without using the framework. Similar to what we have introduced before, we consider the following KGE models.

- DistMult [48]: The bilinear-diagonal embedding model, which uses diagonal matrix to represent the relation between head and tail entities.
- SimplE [24]: The model is a simple enhancement of the Canonical Polyadic (CP) decomposition model [20] by learning two dependent embeddings for each entity.
- ComplEx [37]: The model learns KG embedding in a complex number space, which uses complex number vectors to represent the entities and relations.
- TransE [6]: The translational embedding model, which minimizes the distance between head and tail entities after translation by the relation vector.

All of the models are implemented by PyTorch, an open source library. And we use the baselines implemented by OpenKE[2] [19], an open source tool-kit for knowledge graph embedding.

5.1.3 ***Evaluation Metrics.*** For each triplet (h, r, t) in the testing set, we first use h and r to rank all tail entities, and evaluate the

[1] Source code available at https://github.com/rutgerswiselab/NS-KGE.

[2] https://github.com/thunlp/OpenKE

Dataset	FB15K237					WN18RR				
Metric	MRR ↑	MR ↓	HR@10 ↑	HR@3 ↑	HR@1 ↑	MRR ↑	MR ↓	HR@10 ↑	HR@3 ↑	HR@1 ↑
DistMult	0.177	430.15	0.345	0.198	0.010	0.320	**4019.98***	0.461	0.371	0.240
NS-DistMult	**0.227***	**361.61***	**0.373***	**0.248***	**0.155***	**0.411***	7456.29	**0.462**	**0.424***	**0.384***
SimplE	0.183	387.90	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***
ComplEx	0.240	529.42	**0.415***	**0.271***	0.151	0.390	**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***
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

Table 3: Result on prediction accuracy. NS-X means the non-sampling version of model X under our NS-KGE framework. ↑ means the measure is the higher the better, while ↓ means the measure is the lower the better. Bold numbers represent better performance, and * indicates its performance is significantly better at $p < 0.01$ than the other model.

position of the correct tail entity t. And then we use t and r to rank all head entities, and evaluate the position of the correct head entity h. As a result, if there are $|S|$ triplets in the testing set, we will conduct $2|S|$ evaluations.

We use Hit Ratio (***HR***), Mean Rank (***MR***) and Mean Reciprocal Rank (***MRR***) to evaluate the models. HR is used to measure whether the correct entity is in the Top-K list. MR is the mean of the correct entity's rank, defined as $MR = \frac{1}{2|S|}\sum_{(h,r,t)\in S}(rank_h + rank_t)$. MRR is defined as $MRR = \frac{1}{2|S|}\sum_{(h,r,t)\in S}(\frac{1}{rank_h} + \frac{1}{rank_t})$. These three metrics are widely used in KG embedding evaluation [6, 24, 28]. For HR and MRR, larger value means better performance, and for MR, smaller value means better performance.

5.1.4 *Parameter Settings*.

We set the default embedding dimension as 200, the number of training epochs as 2000, initial learning rate as 0.0001, and use Adam optimizer [25] for all models. To avoid over-fitting, we apply ℓ_2 normalization over the parameters for all models, and we conduct grid search to find the best coefficient of regularization for each model in $[10^{-1},\ 10^{-2},\ 10^{-3},\ 10^{-4}]$. We also conduct grid search to find the best learning rate decay for each model in $[0.1,\ 0.3,\ 0.5,\ 0.7]$.

For negative sampling-based models, we set the number of negative samples as 25; the batch size is 4000. For non-sampling models, we do not split training data into batches, because our model has lower space complexity; the coefficient of positive instances c^+ is set to 1, and the coefficient of negative instances c^- is grid searched in $[10^{-1}, 10^{-2}, 10^{-3},\ 10^{-4},\ 10^{-5},\ 10^{-6}]$. The default setting of c^- is 0.001 in all experiments except when we are tuning c^- to see its influence. For each model on each dataset, we run the model 5 times and report the average result of the 5 times. We use paired t-test to verify the significance of the results.

5.2 Performance Comparison

We apply NS-KGE to DistMult, SimplE, ComplEx and TransE. The experimental results on prediction accuracy are shown in Table 3, and more intuitive comparison are shown in Figure 1. We have the following observations from the results.

First and most importantly, compared to the four baselines, in most cases, our NS-KGE framework achieves the best performance on both of the two datasets. Although some baselines are slightly better than NS-KGE in some cases, for example, on the WN18RR dataset, SimplE's HR@10 has a slightly better performance, but we can see the results are comparable. For HR@1 and HR@3, NS-KGE has 9.78% and 49.01% improvement on average, respectively.

We also conducted some qualitative analysis of the entity ranking results, as shown in Table 4. First, for the same entity and relation, we see that the correct prediction gains higher rank in our non-sampling models. Second, compared to the sampling-based model, the top-10 ranked entities by our non-sampling model tend to be intuitively more relevant to the given entity and relation.

The reason why applying the NS-KGE framework can improve the performance of sampling-based methods (DistMult, SimplE, Complex and TransE) is that, the sampling-based methods only use part of the negative instance information in the dataset, and these models may ignore some important negative instances. However, our NS-KGE framework makes it possible to use all of the information in the dataset and brings better computational time and space consumption at the same time(to be discussed in the following experiments). Therefore, NS-KGE can avoid the problem of sampling-based methods and thus improve the performance.

One interesting observation is that on the WN18RR dataset, our NS-KGE framework is consistently better on the MRR measure, but is worse on the MR measure. The difference between MRR and MR is that MR is more sensitive to bad cases. Due to the large number of entities in the WN18RR dataset, if the correct entity is ranked to lower positions in some lists, it will have a huge influence on the MR measure, but not too much on the MRR measure due to the reciprocal operation. The result implies that our framework may rank the correct entity to very low positions in some cases. However, since our performance on MRR is better, it means that in most cases our framework ranks the correct entity to top positions.

Besides, the observation that NS-KGE improves the performance of both factorization-based (DistMult, SimplE, ComplEx) models and translation-based (TransE) models indicates the effectiveness of the NS-KGE framework, and also shows the potential of applying the framework on other KG embedding models.

5.3 Analyses of Hyper Parameters

In this section, we will analyze the impact of different dimension size d and the negative instance weight c^-.

Dataset	FB15K237			
Relation	media common/netflix genre/titles			
Entity	The Notebook		Funny Girl	
Model	NS-DistMult	DistMult	NS-SimplE	SimplE
Predicted Top-10 Entities	comedy	comedy	comedy	United States of America
	historical period drama	drama	**historical period drama**	drama
	fantasy	thriller	fantasy	romance film
	drama	romance film	drama	thriller
	biography	musical film	musical film	psychological thriller
	thriller	fantasy	biography	DVD
	musical film	suspense	political drama	**historical period drama**
	psychological thriller	**historical period drama**	psychological thriller	crime fiction
	suspense	mystery	thriller	mystery fiction
	mystery	United States of America	suspense	United Kingdom

Table 4: Qualitative results on entity ranking. NS-X means the non-sampling version of model X under our NS-KGE framework. Bold entities are the ground truth. Common entities between model X and NS-X are in gray to highlight the difference entities.

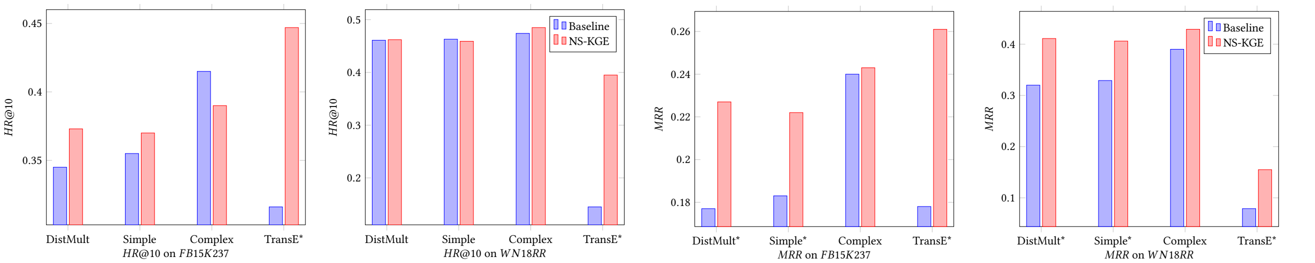

Figure 1: Performance on HR@10 and MRR with and without the NS-KGE framework. The models on x-axis labeled with "*" mean that the performance of NS-KGE framework improved more than 20% from the corresponding baselines.

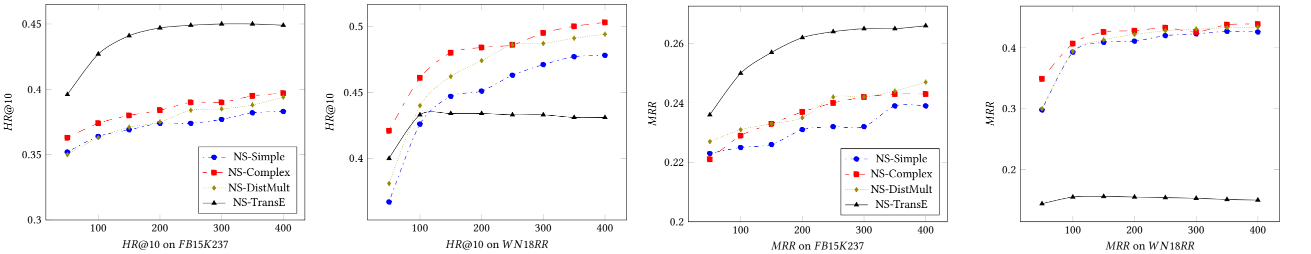

Figure 2: Performance on HR@10 (left two figures) and MRR (right two figures) under different dimension size d.

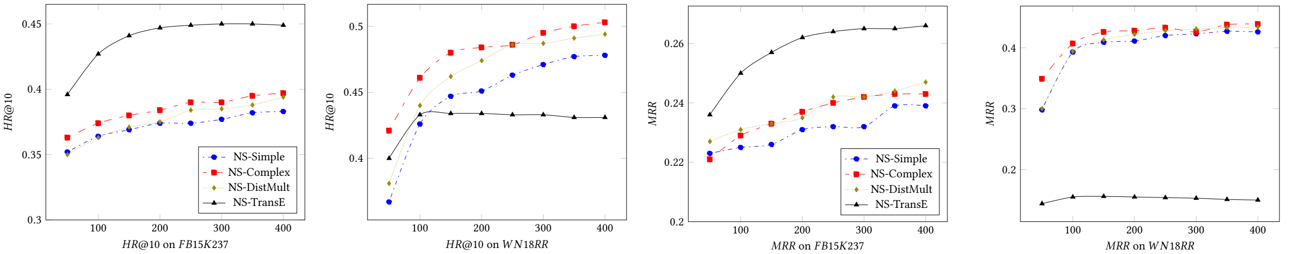

Figure 3: Performance on HR@10 (left two figures) and MRR (right two figures) under different negative instance weight c^-.

Impact of dimension size. Figure 2 shows the performance of our NS-KGE framework under different choices of embedding dimension size d. We can see that in most cases, the performance becomes better when the embedding dimension size increases. It indicates that higher model expressiveness power contributes to better performance in the NS-KGE framework. However, a larger dimension size can also cost more computing time in model training. As we have shown in Section 3.3, the training time is proportional to the square of dimension size. Therefore, we need to trade-off between the training time and the performance. As we can see in

Figure 2, in most cases, the performance tends to be stable at round 200 dimensions. As a result, we choose 200 as the default dimension size for all of the models.

Impact of negative instance weight. In this experiment, we fix the positive instance weight in the NS-KGE framework as $c^+ = 1$, and we tune the negative instance weight c^- to analyze its influence. Figure 3 shows the results when we change the negative instance weight c^- on the two datasets. We see that for all models on both datasets, when the value of c^- increases, the performance tends to increase first and then decreases when c^- is too large. This shows that a proper selection of c^- value is important to the model performance. If c^- is too small (e.g., close to 0), the model would not be able to leverage the information included in the negative instances of the KG. However, the information in negative instances is also noisy, e.g., if two entities are not connected, this may not directly indicate they are irrelevant, instead, this may be caused by the noise in data collection process. As a result, negative instance information is not as reliable as positive instances, and if c^- is too large, it may decrease the performance. Because negative samples are usually much more than positive samples, to avoid class imbalance, the weight of negative instances c^- should be smaller than c^+. In most cases, the optimal selection of c^- is 0.001.

5.4 Efficiency Analyses

In this section, we will discuss the training efficiency of our NS-KGE framework. We will compare the training time of the four sampling-based models DistMult, SimplE, Complex, TransE and their non-sampling versions. For fairness of comparison, all experiments run on a single NVIDIA Geforce 2080Ti GPU. The operating system is Ubuntu 16.04 LTS. For all models, the embedding dimension is set as 200 and number of training epochs is 2000. Results on model training time is shown in Table 5.

	FB15K237	Speed-up	WN18RR	Speed-up
DistMult	3546s	1.00	1922s	1.00
NS-DistMult	**53s**	66.91	**57s**	33.72
SimplE	4447s	1.00	2450s	1.00
NS-SimplE	**73s**	60.92	**77s**	31.82
TransE	2353s	1.00	673s	1.00
NS-TransE	**107s**	21.99	**86s**	7.83
ComplEx	6736s	1.00	3346s	1.00
NS-ComplEx	**157s**	42.90	**158s**	21.18

Table 5: Experimental results on model training time. The models are ordered from top to bottom in ascending order of the training time on each dataset. Speed-up shows how many times NS-X is faster than the corresponding model X.

We can see that the training efficiency of our NS-KGE framework is significant better than the baseline models. For example, if we apply NS-KGE to the DistMult model on the FB15K237 dataset, it only takes 53s to finish training the model, while the original sampling-based DistMult model takes 3546s. The acceleration is about 70 times. For other models and datasets, we also get 20 ∼ 60 times acceleration. This is not surprising because for the sampling-based KGE models, a lot of computational time needs to be spent on sampling the negative examples, while our framework eliminates the sampling procedure. In our implementation, we used in-memory sampling instead of on-disk sampling for the baselines, however, our NS-KGE framework is still much faster than the baselines.

Another intersting observation from Table 5 is that on each dataset, the computational time of our NS-KGE models are NS-DistMult < NS-SimplE < NS-TransE < NS-ComplEx. This is consistent with the mathematical analysis in Section 3 and Section 4. For the NS-DistMult model, its final loss function has one L^A term (see Eq.(6)). For the NS-SimplE model, its final loss function has three L^A terms (Eq.(11)). For the NS-TransE model, its final loss function has four L^A terms (Eq.(14)). While for the NS-ComplEx model, it final loss function has six L^A terms (see Section 4.3, we only show one of terms in Eq.(13)). As we have shown in Section 3.3, the L^A term(s) take the most significant computational time in the loss function. As a result, the final model training time is proportional to the number of L^A terms in the loss function.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposes NS-KGE, a non-sampling framework for knowledge graph embedding, which leverages all of the positive and negative instances in the KG for model training. Besides, we provided mathematical methods to reduce the time and space complexity of the framework, and have shown how the framework can be applied to various KGE models. Experiments on two benchmark datasets demonstrate that the framework is able to enhance both the model performance and the training efficiency.

In the future, we will consider applying our framework on more complex KGE models such as neural network-based models, as well as more complex graph structures such as attributed graphs. We also plan to apply our framework to other graph computation models beyond KGE, such as graph neural networks.

ACKNOWLEDGMENTS

We appreciate the reviews and suggestions of the reviewers. This work was supported in part by NSF IIS-1910154 and IIS-2007907. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W Bruce Croft. 2019. Explainable product search with a dynamic relation embedding model. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2019), 1–29.

[3] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. 2020. Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1995–2010.

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. *Proc. of ACM SIGMOD Int. Conf. on Manage. Data* 1247-1250 (2008).

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[6] A. Bordes, N. Usunier, A. Garcıa-Duran, J. Weston, and O. Yakhnenko. 2013. Translating embeddings for modeling multirelational data. *Adv. Neural Inf. Process. Syst* 2787-7895 (2013).

[7] Liwei Cai and William Yang Wang. 2018. Kbgan: Adversarial learning for knowledge graph embeddings. In *Proceedings of NAACL-HLT 2018*.

[8] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Non-Sampling Factorization Machines for Optimal Context-Aware Recommendation. In *Proceedings of The Web Conference 2020.* 2400–2410.

[9] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Jointly Non-Sampling Learning for Knowledge Graph Enhanced Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 189–198.

[10] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 225–234.

[11] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[12] Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 19–26.

[13] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. 2021. Neural Collaborative Reasoning. In *WWW*.

[14] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. *AAAI* (2018).

[15] Laura Dietz, Alexander Kotov, and Edgar Meij. 2018. Utilizing knowledge graphs for text-centric information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 1387–1390.

[16] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. 2020. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 69–78.

[17] Zuohui Fu, Yikun Xian, Yaxin Zhu, Yongfeng Zhang, and Gerard de Melo. 2020. COOKIE: A Dataset for Conversational Recommendation over Knowledge Graphs in E-commerce. *arXiv preprint arXiv:2008.09237* (2020).

[18] Shijie Geng, Peng Gao, Moitreya Chatterjee, Chiori Hori, Jonathan LeRoux, Yongfeng Zhang, Hongsheng Li, and Anoop Cherian. 2021. Dynamic Graph Representation Learning for Video Dialog via Multi-Modal Shuffled Transformers. In *AAAI*.

[19] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An Open Toolkit for Knowledge Embedding. In *Proceedings of EMNLP*.

[20] Frank L Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics* 6, 1-4 (1927), 164–189.

[21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. 2020. Knowledge graphs. *arXiv preprint arXiv:2003.02320* (2020).

[22] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining.* 105–113.

[23] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388* (2020).

[24] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems.* 4284–4295.

[25] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR* (2015).

[26] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, and S. Auer et al. 2015. DBpedia: A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 167-195 (2015).

[27] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.*

[28] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. 2015. Learning entity and relation embeddings for knowledge graph completions. *Proc. 29th AAAI Conf. Artif. Intell* 2181-2187 (2015).

[29] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *The World Wide Web Conference.* 1210–1221.

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

[31] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data.. In *Icml*, Vol. 11. 809–816.

[32] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining.* IEEE, 995–1000.

[33] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 1–22.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[35] F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. YAGO: A core of semantic knowledge. *Proc. 16th Int. Conf. on World Wide Web* 697-706 (2007).

[36] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality.* 57–66.

[37] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML).

[38] Menghan Wang, Mingming Gong, Xiaolin Zheng, and Kun Zhang. 2018. Modeling dynamic missingness of implicit feedback for recommendation. In *Advances in neural information processing systems.* 6669–6678.

[39] Pengfei Wang, Hanxiong Chen, Yadong Zhu, Huawei Shen, and Yongfeng Zhang. 2019. Unified collaborative filtering over graph embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 155–164.

[40] Peifeng Wang, Shuangyin Li, and Rong Pan. 2018. Incorporating gan for negative sampling in knowledge representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[41] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* (2017).

[42] Q. Wang, Z. Mao, B. Wang, and L. Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[43] Z. Wang, J. Zhang, J. Feng, and Z. Chen. 2014. Knowledge graph embedding by translating on hyperplanes. *Proc. 28th AAAI Conf. Artif. Intell* 1112-1119 (2014).

[44] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 285–294.

[45] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. 2020. CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* 1645–1654.

[46] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web.* 1271–1279.

[47] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.*

[48] Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015* (proceedings of the international conference on learning representations (iclr) 2015 ed.).

[49] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining.* 353–362.

[50] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval.* 785–788.

[51] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101. https://doi.org/10.1561/1500000066

[52] Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019. NSCaching: simple and efficient negative sampling for knowledge graph embedding. In *2019 IEEE 35th International Conference on Data Engineering (ICDE).* IEEE, 614–625.

[53] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving conversational recommender systems via knowledge graph based semantic fusion. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1006–1014.