Simple negative sampling for link prediction in knowledge graphs

Md Kamrul Islam, Sabeur Aridhi, and Malika Smail-Tabbone

Universite de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France,
{kamrul.islam, sabeur.aridhi, malika.smail}@loria.fr

Abstract. Knowledge graph (KG) embedding methods learn the low dimensional vector representations of entities and relations of a knowledge graph, facilitating the link prediction task in knowledge graphs. During learning of embeddings, sampling negative triples is important because KGs have only observed positive triples. To the best of our knowledge, uniform-random, generative adversarial network(GAN)-based, and NSCaching, structure aware negative sampling(SANS) are four negative sampling methods in the literature. Unfortunately, they suffer from computational and memory inefficiency problems. In addition, their prediction performance are affected by the 'vanishing gradient' problem because of poor quality of sampled negative triples. In this paper, we propose a simple negative sampling (SNS) method based on the assumption that the entities which are closer in the embedding space to the corrupted entity are able to provide high-quality negative triples. Furthermore SNS has a good exploitation potential as it uses sampled high-quality negatives for improving the quality of negative triples in next steps. We evaluate our sampling method through link prediction task on five well-known knowledge graph datasets, WN18, WN18RR, FB15K, FB15K-237, YAGO3-10. The method is also evaluated on a new biological KG dataset (FIGHT-HF-23R). Experimental results show that the SNS improves the prediction performance of KG embedding models, and outperforms the existing sampling methods.

Keywords: knowledge graph embedding, link prediction, negative sampling

1 Introduction

A knowledge graph (KG) is a graph-based representation of knowledge which illustrates real-world entities and their relations covering various domains [1]. Formally, a KG is represented as a collection of RDF triples, (head,relation,tail) where the head and the tail are two entities which are connected by a specific relation, e.g.(Shakespeare, isAuthorOf, Hamlet). KGs are fundamental building blocks for many applications, ranging from question answering to content-based recommendation. Some of the notable KGs are FreeBase, WikiData, DBPedia, Yago, NELL. Generally, KGs contain millions of entities and billions of triples.

Despite of their huge size, the incompleteness of KGs is well known. For example, birth place of more than 70% of person entities in Freebase is missing [2]. The incompleteness issue of KGs motivates researchers to study on how to add new triples in KGs. This task is known as link prediction which infer new triples based on the observed triples in KGs.
In recent years, many researchers have developed embedding models to learn vector representations (or embeddings) entities and relations in KGs. These models perform the link prediction task based on the learned embeddings. The training of these models requires positive triples as well as negative/non-observed triples. However, only positive triples are available in KGs. Importantly, the quality of negative triples does matter [3, 4]. This statement brings the importance of sampling negative triples. Unfortunately, this important perspective of embedding model is less focused in the literature. One way to sample negatives is designed based on 'closed-world' assumption where all of the non-observed triples are necessarily false and used as negatives. However, this assumption is not entirely true for KGs due to their incompleteness [5]. Alternatively, most of the KG embedding models samples negatives from non-observed triples under 'open-world' assumption where a non-observed triple may be positive or negative. To the best of our knowledge, there exist four negative sampling methods: uniform-random [6], GAN-based [3, 7, 8], NSCaching [4], and SANS [9] . However, each of them has its own pros-cons and the current state-of-art still lacks a good negative sampling method. In this article, we propose a simple but efficient method called SNS to sample high-quality negative triples in KG. In sampling, the trade-off between exploration and exploitation is crucial in searching for a high-quality samples [10]. Exploration corresponds to the capacity of the sampling method to select high-quality negative triples from unexplored areas whereas exploitation favours the utilization of already known negative triples to sample other negatives. SNS makes a good balance between exploration and exploitation to improve the quality of negative triples. We designed SNS as general sampling method that can be plugged to any KG embedding model for link prediction.

2 State-of-art

High-quality negative triples, which are not readily available, contribute during training of a KG embedding model. Generally, candidate negative triple set is generated by positive triple perturbation where head/tail entities are replaced with other entities [5] and then a sampler samples negative triples from the set. 'Uniform-random' is the mostly used negative sampling method where negative triples are randomly sampled from a uniform distribution of candidate negatives [4]. Though 'uniform-random' is simple, a sampled negative triple could be completely unrelated to the corresponding positive triple and is easily classified as negative. Consequently, 'uniform-random' method seriously suffers from 'zero-loss'/'vanishing- gradient' problem [3]. Recently, the generative property of GAN frameworks to generate high-quality negatives for avoiding the 'vanishing-gradient' problem is studied. The generator of GAN is trained to pick high-

quality negative triples whereas the discriminator part is trained to learn embeddings. IGAN [8], KBGAN [3], KSGAN [7] are three existing GAN-based sampling methods for KGs. Compared to 'uniform-random' sampling, these methods improve the quality of negative triples undoubtedly. However, these methods increase the number of model parameters and take extra costs on training for parameter optimization [4]. In addition, they suffer from instability and degeneracy problems because of adopting the complex reinforcement learning to train the generator [4]. To avoid the excessive training time of GAN-based methods, Zhang et al [4] proposed a 'distilled' version of GAN-based methods, namely NSCaching which stores negatives with high scores in head and tail caches for each positive triple, and then samples negatives directly from the caches. With NSCaching sampling, KG embedding models show competitive link prediction performance. However, the memory requirement increases exponentially with the size of KGs. Also, the regular updating of caches increases the computational time. Thus, scalablity is a big issue for NSCaching and it is not recommended for large KGs. Apart from the above-mentioned sampling methods, the hard negative mining in contrastive learning motivates Ahrabian et al. [9] to study the neighborhood information of entities to sample negatives. They develop SANS method based on an assumption that neighbor entities without direct relation are good candidates for generating negatives. However, the one-time generation of negatives prior to the start of the embedding learning process is expensive in terms of memory requirements, as it requires the whole adjacency matrix of a KG in main memory.

3 The proposed SNS method

For link prediction, KG embedding models are trained with positive and negative triples to learn embeddings of entities and relations. In this section, we first briefly describe a classical KG embedding model, and then our proposed SNS negative sampling method. Throughout the paper, we use $\mathbb{E}$ to denote the set of all entities, $\mathbb{R}$ to denote the set of all relations, $\mathbb{S}$ to denote the set of positive training triples, $\mathbb{D}$ to denote the set of positive test triples, $\mathbb{Q}$ to denote the set of all positive triples, d to denote embedding dimension size, m to denote the batch size, S_m and S'_m to denote a batch of positive and respective negative triples. The architecture of a classical KG embedding model is given in Fig. 1 which starts with initializing the embeddings of entities and relations randomly from uniform/Gaussian distributions [5]. For the training of the model, a batch of positive train triples S_m is fetched from the train triple set, $\mathbb{S}$. A negative sampling method is then used to generate a batch of negative triples. In the architecture, we inject our SNS method(shaded by yellowish color in Fig. 1) to generate the batch of negative triples, S'_m for the batch of positive triples,S_m. The batches of positive and negative triples are then used by the pairwise training strategy to learn the embeddings. In pairwise training, the model tries to assign more plausibility score to a positive triple than its corresponding negative triple. The training objective is to optimize the embeddings of entities and relations

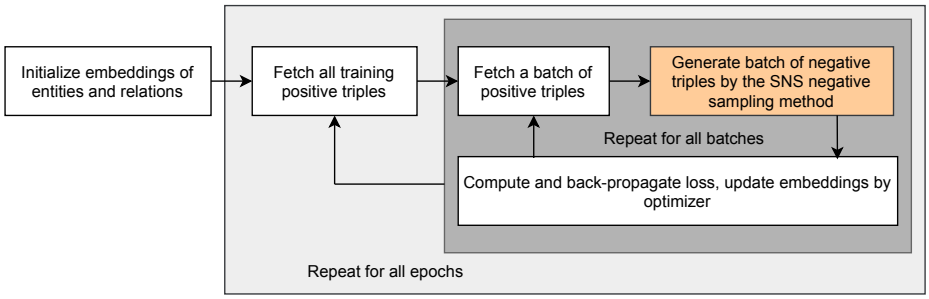


Fig. 1: Architecture of a classical KG embedding model with SNS sampling

for minimizing the total pairwise loss as designed Equation 1.

$$\min_{\Theta} \sum_{\forall_{(h,r,t)\in S_m,(h',r,t')\in S'_m}} L(f_r(h,t), f_r(h',t')) + \lambda reg(\Theta) \tag{1}$$

Here, f_r is the scoring function of the embedding model, $(h,r,t) \in S_m$ is a positive triple and $(h',r,t') \in S'_m$ is the corresponding negative triple. The pairwise loss for the positive and its negative triple is defined in Equation 2 [5].

$$L(f_r(h,t), f_r(h',t')) = \Big[\lambda - f_r(h,t) + f_r(h',t')\Big]_+ \tag{2}$$

Here, λ is the margin and $[.]_+ = max(0,.)$ is the hinge function. The embedding updating process is repeated for all batches of positive triples(shaded by dark gray color in Fig. 1), and the whole training process (shaded by light gray color in Fig. 1) is repeated for T times (or epochs). The model training process is similar to a traditional KG embedding model except we adapt our negative sampling method. We refer to [5] for more details about the traditional KG embedding.
In the following, we describe our proposed SNS method for negative sample generation. The SNS method aims to generate high-quality negative triples for avoiding the 'vanishing-gradient' problem of uniform-random sampling, the complex parameter optimization problem of GAN-based sampling and the excessive memory requirement problems of NSCaching. Fig. 2a shows the basic steps of SNS sampling. The steps are described in the following.

Step 1. Triple perturbation: From Fig. 2a, SNS starts with generating a initial negative set for a positive triple by positive triple perturbation. This step is similar to other sampling methods. In triple perturbation, the head/tail of the positive triple is corrupted by replacing head/tail with other entities in the entity set($\mathbb{E}$). In the same time, it is checked that the negative set does not contain any positive triple. To illustrate, consider the positive triple $q = (h,r,t) \in \mathbb{S}$. Corrupting the tail(t) gives the initial negative set $q'_0(t) = \{(h,r,t') \notin \mathbb{Q} | t' \in \mathbb{E}\}$.

Step 2. Candidate set generation: Generally, the size of the set $q'_0(t)$ is large as KG contains large number of entities. Zhang et al [4] describe that only

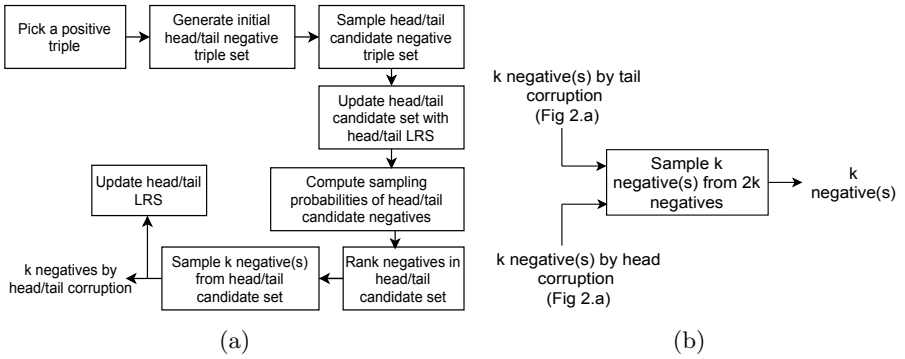


Fig. 2: The SNS sampling method (a) generation of k high-quality negatives by corrupting head/tail, (b) k negative(s) sampling from 2k high-quality negatives

some of initial negatives of the set are of good-quality. As we need few negatives for each positive triple, we randomly sample N_1 triples from $q'_0(t)$ to generate candidate negative set $q'_1(t)$(i.e. $q'_1(t) \subseteq q'_0(t)$) for user defined parameter N_1.
As the training progresses, it is desired that quality of the negative(s) for the next step will be better or close to the quality of negative(s) in the current step. For this purpose, the sampled negatives in the current step are stored in a small structure called least recently selected(LRS). $LRS[q'_h, q'_t]$ saves sampled negatives where q'_t and q'_h are the sampled negatives by head and tail corruption respectively in the previous step. The saved negatives are used in the next step for sampling high-quality negatives. The candidate negative set, $q'_1(t)$ is updated to include the LRS negative(s) as $q'_1(t) := q'_1(t) \cup LRS[q'_t]$. The use of LRS is intended to favour the exploitation behavior of the proposed SNS method. It ensures that the quality of the negative(s) at current step is better than or at least close to the quality of the negative(s) at the previous step.

Step 3. Sampling probability computation: In this step, we compute the sampling probability of each negative in the candidate negative set $q'_1(t)$. Negatives with higher probabilities are considered as high-quality negatives. The probability is defined based on the distance score of each negatives. The distance score for the negative triple, $(h, r, t'_i) \in q'_1(t)$ is computed as the distance between the corrupted(t) and the new entities(t') as Equation 3.

$$d(t, t'_i) = ||\mathbf{t} - \mathbf{t'_i}|| \tag{3}$$

Here, $\mathbf{t}$ and $\mathbf{t'_i}$ are embeddings of the entities t and t'_i. A softmax function is then used to compute the sampling probability score of each candidate negative $(h, r, t'_i) \in q'_1(t)$ as Equation 4.

$$P(h, r, t'_i) = \frac{exp(\frac{1}{d(t,t'_i)})}{\sum_{j=1}^{N_1} exp(\frac{1}{d(t,t'_j)})} \tag{4}$$

The softmax function computes higher sampling probability for the candidate negative triples with lower distance score.

Step 4. Negative triple sampling and LRS updating: The triples from the candidate negative set,$q'_1(t)$ are ranked in decreasing order of their probabilities and k negative(s) are sampled. A natural choice could be sampling top-k (k=1 for pairwise training, $k > 1$ for maximum likelihood training) negative(s). However, sampling top-k ranked negative(s) could arise two problems. Firstly, there is a chance of high repeat sampling of the same negative(s) (even in many consecutive steps) as the current candidate negative set also includes the least recently sampled (LRS) high-quality negative(s). This case affects the exploration of SNS sampling. Secondly, the existence of false negative triples (looks like high-quality) is not ignorable [5]. To tackle these problems, SNS randomly samples k negative triples from N_2 top ranked triples in q'_t as $q'_t = \{(h, r, t')|rank_{(h,r,t')\leq N_2}\}$ for user defined parameter N_2 where $N_2 > k$.
SNS repeats the above described process (from initial negative set generation to k negative(s) sampling) for head(h) corruption to sample k negative(s) as q'_h for the positive triple, q. The $LRS[q't, q'h]$ is updated with the sampled 2k high-quality negative(s). Finally, we randomly sample k high-quality negative(s) as $q'_{h,r,t}$ from 2k negatives in $q'_h \cup q'_t$. The sampled k negative(s), q' and the corresponding positive, q are then used to train the KG embedding model.

4 Experiments

To evaluate the efficiency of SNS sampling, we plug it to a KG embedding model. KG embedding models are broadly categorized into two main categories: (1) translational models and (2) semantic matching models [5]. Translational models consider each relation as a translation in the embedding space: adding relation to the head gives a close position to the tail [6]. On the other hand, semantic matching models define the plausibility of a triple by matching representations of entities and relations embodied in their embeddings [5]. The main focus of this paper is negative sampling for KGs. For evaluation, we choose TransH [11] to represent translational and DistMult [12] to represent semantic matching model as they are popular baselines for link prediction task in KGs. For details about the models, we refer to the original papers. The scoring functions of the models are given in Table 1. Each of the model is evaluated for three types of existing (i.e. random, GAN-based, NSCaching) and for the proposed SNS methods. We do not include SANS in the experiment due to its excessive memory requirement. For GAN-based method, we choose KBGAN as it is the only GAN-based sampling which has publicly available implementation (to best of our knowledge). We refer to the original articles for details about the sampling methods.

4.1 Datasets

In the experiments, we use five widely used benchmark KG datasets, i.e., FB15K, FB15K-237, WN18, WN18RR, YAGO3-10 for link prediction task [6, 3, 4]. WN18

Table 1: Scoring functions of KG embedding models: w_r is the normal vector of the hyperplane for the relation r, $diag(r)$ is the diagonal matrix for the relation r, and $||.||_2$ represents l_2 norm.

Model	**Embeddings**	**Scoring function,** $f_r(h,t)$
TransH [11]	$\mathbf{h}^d, \mathbf{t}^d, \mathbf{r}^d, w_r^d$	$\|\|(\mathbf{h} - w_r^T \mathbf{h} w_r) + \mathbf{r} - (\mathbf{t} - w_r^T \mathbf{t} w_r)\|\|_2^2$
DistMult [12]	$\mathbf{h}^d, \mathbf{t}^d, \mathbf{r}^d$	$\mathbf{h^T} diag(r) \mathbf{t}$

Table 2: The experimental KG datasets

KG datasets	**#Entity**	**#Relation type**	**#Facts**	**#Train**	**#Valid**	**#Test**
WN18RR	40,943	11	93,003	86,835	3,034	3,134
WN18	40,943	18	151,442	141,442	5,000	5,000
FB15K-237	14,541	237	310,116	272,115	17,535	20,466
FB15K	14,951	1,345	592,213	483,142	50,000	59,071
FIGHT-HF-23R	90,430	23	948,298	853,482	47,402	47,414
YAGO3-10	113,273	37	1,089,040	1,079,040	5,000	5,000

is derived from the WordNet which is a large semantic lexicon for the English language. FB15K is a subset of triples from Freebase KG which is a large collaborative general knowledge base. WN18RR and FB15K-237 datasets are derived from WN18 and FB15K respectively after removing the inverse-duplicate relations. The YAGO3-10 dataset is extracted from the open source YAGO knowledge base considering the entities with at least 10 relations.

We also provide a biological KG dataset, namely FIGHT-HF-23R. FIGHT-HF is a biological knowledge graph which describes named relations among several biological types such as proteins, genes, diseases, drugs [13]. The original graph contains 246,672 biological entities and 24,601,110 relations of 37 types. We extract a medium size dataset by considering only those relations which have more than 50 facts and less than 500K facts. As a result, the dataset, naming FIGHT-HF-23R, contains 90,430 entities, 23 relation types and 948,298 triples. An example of extracted triple is (Cyclosporin A, drug_indication, Pterygium). The details about the dataset are available in GitLab repository[1].

The KG datasets, except the new FIGHT-HF-23R, come with train/valid/test splits. For FIGHT-HF-23R dataset, we split the dataset into train, validation, test triples. Unfortunately, we do not find any standard rule for splitting KG dataset. We split the dataset into 90/5/5 for train/validation/test triples so that we have enough triples to train the model. We apply the split rule to relation label where positive triples (facts) with a specific relation are also split with 90%/5%/5% ratio. This split ensures that we have all types of relations in all splits. We also check that no isolated entities exist in the training dataset as in this case the entities will have low quality embedding. The split gives train, test and valid triple sizes as in Table 2.

[1] https://gitlab.inria.fr/kislam/sns

4.2 Evaluation metrics

To evaluate the performance of any sampling method, we plug it to a KG embedding model for link prediction task. The performance is defined with two widely used metrics: Hit@z, and mean reciprocal rank (MRR) [14]. The metrics are defined based on the rank of the positive test triple. Hit@z is defined as the average number of times a positive test triple is among the z highest ranked triples; whereas MRR is the average reciprocal rank of the positive test triple [15]. The range of both scores is 0 to 1. The higher value of MRR demonstrates the better ranking of positive test triples and better ranking provides better prediction performance. Also, higher Hit@z score indicates better performance. To illustrate, consider the positive test triple $q = (h, r, t) \in \mathbb{D}$. A set of negative triples $q'_t = \{(h, r, t') \notin \mathbb{Q} | t' \in \mathbb{E}\}$ is generated by simple triple perturbation (replacing tail with other entities) [6, 16] confirming that no positive triple exists in q'_t. The triples in $q \cup q'_t$ are then ranked in decreasing order of their scores (computed by embedding-based scoring functions in Table 1). The rank of the positive test triple q in $q \cup q'_t$ is defined as $rank^t_q$. Based on the rank of each positive test triple q, the performance metrics are defined in Equations 5 and 6.

$$Hit^t@z = \frac{1}{|\mathbb{D}|} \sum_{q \in \mathbb{D}} hit^t_q, \;\; hit^t_q = \begin{cases} 1, & if \; rank^t_q \leq z \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$MRR^t = \frac{1}{|\mathbb{D}|} \sum_{q \in \mathbb{D}} \frac{1}{rank^t_q} \tag{6}$$

The whole evaluation process is also repeated by corrupting the head entity of each positive triple as well and $Hit^h@z$, MRR^h are computed. The final $Hit@z$, MRR metrics are the average of head and tails metrics i.e. $Hit@z = (Hit^t@z + Hit^h@z)/2$ and $MRR = (MRR^t + MRR^h)/2$. We re-scale the Hit@z score from the range 0-1 to 0-100 to facilitate the comparisons. As suggested by the most of the literature for link prediction in KGs, we consider $z \in \{1, 3, 10\}$.

4.3 Results and discussion

The proposed SNS method is implemented in Python with well-known PyTorch and run on a 'NVIDIA A100-PCIE-40GB' GPU. We set training epoch number to 200, embedding dimension to 100, learning rate to 0.0001, margin value to 4.0 for all the experiments. For the embedding optimization task, we use the popular Adam optimizer[17]. We evaluate link prediction performance of embedding models with SNS sampling, and compare to other sampling methods in six KG datasets. Then, we do more analyses including parameter sensitivity, performance in different epochs for the WN18RR dataset. The relation-wise prediction performance and examples of sampled negatives for WN18RR dataset are available in the same GitLab link as mentioned in Section 4.1.

Prediction Performance: We train the KG embedding models from scratch for each of the sampling methods, Random-uniform, KBGAN, NSCaching, SNS.

For parameter setting, we follow the recommendations from the original paper ([4] for NSCaching, [3] for KBGAN). The prediction performance metrics are tabulated in Table 3. Considering translational model(TransH), undoubtedly the proposed SNS sampling method shows the best prediction metrics among all the negative sampling methods with respect to most of prediction metrics in most of the KG datasets. In WordNet KG datasets (WN18RR, WN18), NSCaching is the second best methods and the performance improved by 2-5% for SNS sampling considering Hit@k metrics when they are used with TransH. We find the best results of SNS sampling for FB15K dataset where the Hit@k scores are improved by 5-8% comparing to second best NSCaching sampling method. For rest of two datasets, SNS also remains best or second best in almost all metrics. Good balance between exploration and exploitation could be the most suitable reason behind this success. When the sampling methods are used in DistMult model, the hit@10 and hit@3 scores drop in all datasets except FIGHT-HF-23R. Training the model for more epochs might improve the metrics. However, our intention is not to compare different prediction models, rather different sampling methods. We see nearly similar trend of improvement in prediction performance for DistMult with SNS sampling. Surprisingly, the performance metric on the new FIGHT-HF-23R KG dataset are poor for TransH with the sampling methods, but better for DistMult with the sampling methods. In the following, we describe further analysis of prediction by TransH with different sampling methods in the smallest KG dataset, WN18RR.

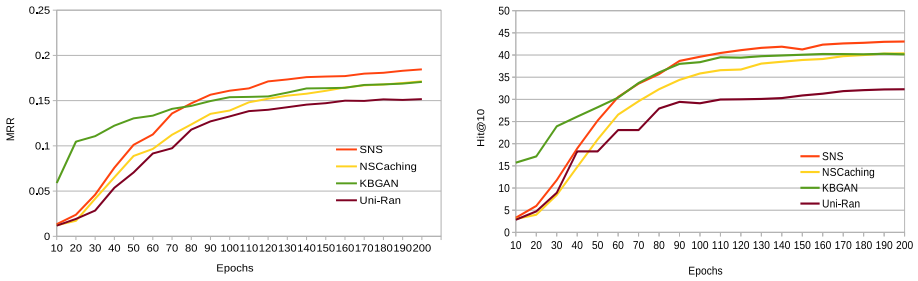
Change in prediction performance for different epochs: To illustrate how the prediction performance changes with varying number of training epoch, we plot the MRR and Hit@10 scores of a embedding model with different sampling methods from epoch 10 to 200 with a interval of 10 in Fig. 3. At the initial point of epoch 10, the MRR score of SNS is nearly same (around 0.02) as other sampling methods, except KBGAN which has the highest MRR score (around 0.06). As the training epoch number increases, the embeddings quality are improved and consequently the rise in MRR scores of all methods are seen. The MRR improving rate is seen higher for SNS method. At the epoch 70, the MRR score of SNS is best. Though the improvement rate is not constant, the MRR scores of the proposed SNS method remain highest among all the sampling methods in the following epochs. We see the worst MRR for 'uniform-random' method as the method does not learn to pick high-quality negative triples. These improvements in rank are reflected in Hit@10 scores curves where SNS has the highest Hit@10 scores in later half of training epochs. These improvements in performance prove that our sampling method is able to provide better ranks of test triples than the state-of-art sampling methods.

Parameter sensitivity analysis: SNS sampling method has two parameters, N_1 and N_2. To describe the changes in performance for different values of these parameters, we record Hit@10 and MRR scores for different values of N_1 with fixed $N_2 = 5$ which are plotted in Fig. 4. As the value of N_1 increases, more initial negative triples are explored and the SNS sampling gets better exploration. As a consequence, the prediction performance improves as the value

Table 3: Link prediction(LP) results: MRR, and Hit@z of different negative sampling(NS) methods on KG datasets. The best and second best metrics are marked in bold and underline faces.

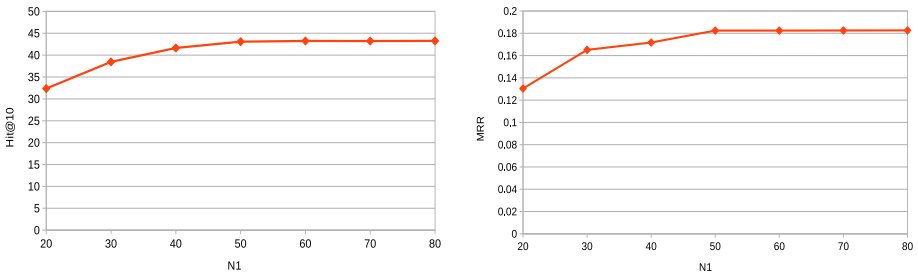
LP models	**NS methods**	**WN18RR**				**WN18**			
		MRR	hit@10	hit@3	hit@1	MRR	hit@10	hit@3	hit@1
TransH	Random	0.1520	32.27	23.72	0.11	0.3199	79.43	64.25	11.85
	NSCaching	0.1713	40.68	31.43	0.93	0.4171	88.65	74.05	17.48
	KBGAN	0.1708	40.08	29.35	0.10	0.4183	87.34	73.67	18.09
	SNS	**0.1852**	**43.04**	**33.82**	**1.80**	**0.448**	**91.47**	**79.30**	**19.28**
DistMult	Random	0.1918	32.16	23.88	14.22	0.3453	52.82	37.33	25.75
	NSCaching	0.2262	37.37	**29.11**	17.84	0.3772	56.85	**42.18**	29.04
	KBGAN	0.2285	33.42	27.23	17.34	0.3791	**57.28**	41.97	28.39
	SNS	**0.2333**	**37.83**	25.12	**18.49**	**0.3931**	56.46	42.13	**31.38**
		FB15K237				**FB15K**			
TransH	Random	0.1988	36.68	22.50	11.50	0.3115	52.88	36.68	19.58
	NSCaching	0.2476	40.39	26.59	**17.33**	0.3926	61.22	44.99	25.50
	KBGAN	0.2162	40.58	23.52	15.23	0.3228	53.67	38.34	20.52
	SNS	**0.2514**	**42.90**	**29.44**	15.18	**0.4360**	**66.72**	**51.52**	**30.58**
DistMult	Random	0.1918	31.82	20.71	12.96	0.2188	33.25	21.84	12.95
	NSCaching	0.2205	34.87	25.69	15.72	0.2327	39.89	27.41	16.19
	KBGAN	0.2282	36.23	**27.23**	13.02	0.2203	35.54	23.12	15.92
	SNS	**0.2471**	**38.18**	26.97	**17.80**	**0.2937**	**45.62**	**32.66**	**20.79**
		YAGO3-10				**FIGHT-HF-23R**			
TransH	Random	0.0850	18.96	9.05	1.24	0.0200	3.95	1.86	0.63
	NSCaching	0.1431	**26.76**	15.97	7.49	0.0294	6.96	1.97	0.66
	KBGAN	0.1467	26.08	16.03	8.34	0.0342	7.24	2.89	0.59
	SNS	**0.1488**	26.72	**16.23**	**8.87**	**0.0410**	**8.92**	2.06	0.64
DistMult	Random	0.0533	10.59	5.44	2.35	0.1439	26.01	14.29	8.75
	NSCaching	**0.0875**	14.22	**8.86**	**5.64**	0.1863	30.36	20.13	13.10
	KBGAN	0.0712	13.98	7.79	3.19	0.1681	28.25	18.64	**14.38**
	SNS	0.0804	**16.72**	8.39	4.98	**0.1899**	**33.78**	**22.19**	12.63

of N_1 increases from 20 to upper as seen from Figs. 4a, 4b. The prediction performance is nearly stable for $N_1 = 50$ and above. In the point $N_1 = 50$, SNS sampling has good exploration to sample sufficient number of high-quality negatives. And this could be the cause of performance stability for $N_1 \geq 50$. Again, to describe the sensitivity of the parameter N_2, we plot the performance metrics by varying N_2 among {1, 2, 3, 4, 5, 6, 7} with fixed N_1=50 in Fig. 5. Figs. 5a and 5b show the change in Hit@10 and MRR for change in N_2. With setting N_2=1, SNS samples the top-most ranked negative(s). In this case, the chance of repeat sampling is high as SNS considers already known high-quality LRS negative(s) in addition to other candidate negatives. In case of high repeat sampling, SNS suffers from low exploration and high exploitation effect leading to drops in MRR and Hit@10 metrics. With N_2=1, we see lowest prediction performance for SNS. As the value of N_2 increases, SNS gets better exploration



(a) Epoch number vs MRR (b) Epoch number vs Hit@10

Fig. 3: Prediction scores of TransH with different samplings in different epochs

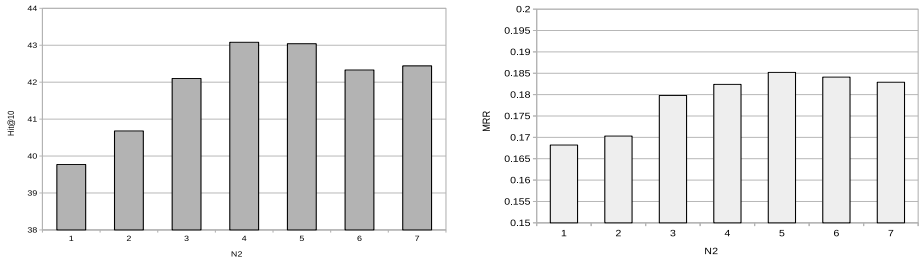


(a) N_1 vs Hit@10 (b) N_1 vs MRR

Fig. 4: Sensitivity of SNS to N_1, size of candidate negative set

and the best balance between exploration and exploitation is found for N_2=5 where the highest prediction metrics are recorded. However as the value of N_2 increases, the chance of sampling of known high-quality triple decreases (poor exploitation) and the chance of sampling less good-quality negative increases. As a result, the performance drops as seen in Fig. 5 where both hit@10 and MRR drop for $N_2 > 5$.

Memory and computational efficiency Undoubtedly, random-uniform is the simplest, fastest, memory efficient sampling method as it does not learn or store any parameter. GAN-based sampling makes the prediction model more complex, increases the number of training parameters, and makes the model harder to train due to use of reinforcement learning [4]. As a consequence, KBGAN needs extra memory and computational cost to store and optimize the parameters. NSCaching stores set of high-quality negative triples in each positive triple cache which makes it worst memory efficient. In addition, the method takes additional time to update the cache periodically. The proposed SNS sampling method does not increase the training parameter like GAN-based method. It memorizes only the least recently sampled negative triple which takes very small amount of memory. Thus, intuitively SNS sampling is more memory ef-



(a) N_2 vs Hit@10 (b) N_2 vs MRR

Fig. 5: Sensitivity of N_2

ficient than GAN-based sampling and NSCaching. The method does not use complex learning method like GAN-based method or does not take extra cache updating time like NSCaching. It takes very small amount of time to update the LRS structure. We record the training time of each sampling method with the TransH embedding model. The data on training time is available at in the same GitLab link as mentioned in Section 4.1. We see that the training time increases as the number of training samples increases, as expected. We find 7-40% and 3-14% improvement in training time for SNS when it is compared to KBGAN and NSCaching respectively.

5 Conclusion

In this paper, we propose a simple and efficient negative sampling method for knowledge graph embedding. The method is general and can be plugged to any knowledge graph embedding method. The method is able to generate high-quality negative triples and takes low computational time and memory while anticipating the 'vanishing-gradient' problem. Experimentally, we evaluate our method on six knowledge graph datasets for link prediction task and also describe its parameter sensitivity. The results show that the proposed SNS sampling brings consistent improvements in prediction performance.
The poor performance of the studied models on YAGO3-10 and FIGHT-HF-23R datasets leaves the future work to explore other embedding models in the literature on these datasets. Implementing the models in a distributed environment to improve computational efficiency is another potential perspective of this work.

References

1. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web, 8(3), pp. 489-508 (2017).
2. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., ...: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the

20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601-610 (2014).
3. Cai, L., Wang, W. Y.: KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1470-1480 (2018).
4. Zhang, Y., Yao, Q., Shao, Y., Chen, L.: NSCaching: simple and efficient negative sampling for knowledge graph embedding. In: 2019 IEEE 35th International Conference on Data Engineering, pp. 614-625 (2019).
5. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering, 29(12), pp. 2724-2743 (2017).
6. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, vol. 26, pp. 1-9 (2013).
7. Hu, K., Liu, H., Hao, T.: A knowledge selective adversarial network for link prediction in knowledge graph. In: CCF International Conference on Natural Language Processing and Chinese Computing, pp. 171-183. Springer, Cham (2019).
8. Wang, P., Li, S., Pan, R.: Incorporating gan for negative sampling in knowledge representation learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, (2018).
9. Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W. L., Bose, A. J.: Structure Aware Negative Sampling in Knowledge Graphs. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pp. 6093-6101 (2020).
10. Chen, J., Xin, B., Peng, Z., Dou, L., Zhang, J.: Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 39(3), pp. 680-691 (2009).
11. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28(1), pp. 1112-1119 (2014).
12. Yang, B., Yih, W. T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: International Conference Learning Representation (2014).
13. Bresso, E., Lacomblez, C., Pizard, A., Rossignol, P., Zannad, F., Smaïl-Tabbone, M., Devignes, M. D.: A data science approach for exploring differential expression profiles of genes in transcriptomic studies-Application to the understanding of ageing in obese and lean rats in the FIGHT-HF project. In: JOBIM 2018-Journées Ouvertes Biologie, Informatique et Mathématiques (2018).
14. Wang, M., Qiu, L., Wang, X.: A Survey on Knowledge Graph Embeddings for Link Prediction. Symmetry, 13(3), p. 485, (2021).
15. Rossi, A., Matinata, A.: Knowledge Graph Embeddings: Are Relation-Learning Models Learning Relations?. In: EDBT/ICDT Workshops, (2020).
16. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, vol. 1, pp. 687-696 (2015).
17. Kingma, D. P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representation, (2015).