Diversified and Adaptive Negative Sampling on Knowledge Graphs

Ran Liu[1], Zhongzhou Liu[1], Xiaoli Li[2], Hao Wu[3]✉, and Yuan Fang[1]✉

[1] Singapore Management University, 81 Victoria St, Singapore 188065
{ran.liu.2020@phdcs.,zzliu.2020@phdcs.,yfang@}smu.edu.sg
[2] Institute for Infocomm Research, 1 Fusionopolis Way, Singapore 138632
xlli@i2r.a-star.edu.sg
[3] Beijing Normal University, 19 Xinwai Ave, Haidian District, Beijing 100875
wuhao@bnu.edu.cn

Abstract. In knowledge graph embedding, aside from positive triplets (*i.e.*, facts in the knowledge graph), the negative triplets used for training also have a direct influence on the model performance. In reality, since knowledge graphs are sparse and incomplete, negative triplets often lack explicit labels, and thus they are often obtained from various sampling strategies (*e.g.*, randomly replacing an entity in a positive triplet). An ideal sampled negative triplet should be informative enough to help the model train better. However, existing methods often ignore *diversity* and *adaptiveness* in their sampling process, which harms the informativeness of negative triplets. As such, we propose a generative adversarial approach called **D**iversified and **A**daptive **N**egative **S**ampling (DANS) on knowledge graphs. DANS is equipped with a two-way generator that generates more diverse negative triplets through two pathways, and an adaptive mechanism that produces more fine-grained examples by localizing the global generator for different entities and relations. On the one hand, the two-way generator increase the overall informativeness with more diverse negative examples; on the other hand, the adaptive mechanism increases the individual sample-wise informativeness with more fine-grained sampling. Finally, we evaluate the performance of DANS on three benchmark knowledge graphs to demonstrate its effectiveness through quantitative and qualitative experiments.

Keywords: Knowledge graphs· Graph representation learning· Graph neural networks· Negative sampling.

1 Introduction

Knowledge graphs have been widely used to encode facts about the real world. Typically, each fact describes a relationship between a head and tail entity in the form of a triplet ⟨head, relation, tail⟩, and different entities across facts are interconnected to form a graph structure. The rich facts contained in a large-scale knowledge graph can be used to enhance numerous applications that rely on real-world knowledge, such as question answering [32, 13], object detection [9] and recommendation [4]. To effectively exploit the facts for these applications, a common approach is to first perform knowledge graph embedding that converts the symbolic entities and relations to a latent vector space, which can then be integrated with other machine learning models.

In this paper, we focus on the problem of *knowledge graph embedding*. The high-level idea is that the embedding vectors of entities and relations co-occurring in the same fact should be bounded by certain constraints due to their relatedness. For instance, consider a fact $\tau = \langle h = \texttt{Beijing}, r = \texttt{isCapitalOf}, t = \texttt{China}\rangle$ and a classic method TransE [2]. TransE maps each entity and relation in the fact to vectors, *i.e.*, $\mathbf{e}_h$, $\mathbf{e}_r$, $\mathbf{e}_t$, respectively, so that they approximately satisfy the constraint $\mathbf{e}_h + \mathbf{e}_r \approx \mathbf{e}_t$ by minimizing the loss $\|\mathbf{e}_h + \mathbf{e}_r - \mathbf{e}_t\|$. On the contrary, a nonfact such as $\langle h = \texttt{Beijing}, r = \texttt{isCapitalOf}, t' = \texttt{Russia}\rangle$ would maximize the loss $\|\mathbf{e}_h + \mathbf{e}_r - \mathbf{e}_{t'}\|$. Given this contrast, the factual triplets are known as *positive* triplets (or examples), whereas the non-factual triplets are called *negative* triplets. Although positive triplets are readily available, negative triplets are often obtained through random sampling. More recent works [34, 3] explore advanced constraints or losses [2, 29] on the triplets, but the sampling strategy for negative triplets remains a crucial yet less explored problem.

Earlier negative sampling approaches resort to random sampling, *e.g.*, by replacing the tail (or head) entity in a positive triplet with a random entity from the knowledge graph sampled in a uniform [33] or popularity-weighted manner [17]. Although random sampling is straightforward, it is often inadequate to optimize the *informativeness* of negative triplets. The informativeness refers to how much information each negative triplet could contribute to model learning. Intuitively, a more informative negative triplet would improve the efficiency of model training and accelerate model convergence. For instance, the positive triplet τ given earlier, ⟨`Beijing`, `isCapitalOf`, `Russia`⟩ is considered a more informative negative triplet than ⟨`Beijing`, `isCapitalOf`, `KFC`⟩, as the latter can be easily identified as negative and thus helps little in refining the decision boundary. Although various scoring functions [2, 29, 35] help to judge the informativeness of negative triplets, they do not consider the *diversity* and *adaptiveness* of the sampling process, which are two aspects we propose to study in this work.

On one hand, diversity helps to increase the *overall* informativeness of all the negative triplets collectively. We observe that negative triplets can be associated with both entities and relations. For example, the tail entity of the positive triplet τ can be replaced by entities associated not only with the head entity `Beijing`, such as `GreatWall` and `Shanghai`, but also with the relation `isCaptialOf`, such as `Russia` (a country with some capital city) and `London` (a capital city of some country). On the other hand, adaptive sampling of negative triplets would make entity- or relation-specific adjustments to sample selection, which increases the *individual* informativeness of each triplet in a finer-grained manner. For instance, selecting a tail entity for `Beijing`, `Tokyo` or `KFC` using a global sampling model could be suboptimal given the variability among these entities. Instead, local models that condition on each entity would be able to adapt to such differences and make each triplet more informative.

In view of the above, we propose a **D**iversified and **A**daptive **N**egative **S**ampling (DANS) approach for knowledge graph embedding, to improve both the overall and individual informativeness of negative triplets. Similar to previous state-of-the-art approaches such as KBGAN [3], we adopt a generative adversarial network (GAN) [30] for the generation of negative samples. However, there are two significant differences from previous GAN-based negative sampling on the knowledge graph. First, we design a two-way generator to produce diversified samples that are associated with both entities and relations w.r.t. a positive triplet, which aims to increase the overall informativeness of the samples. More specifically, the generator consists of two pathways to produce two different kinds of negative triplets associated with a given entity and entity-relation, respectively. Second, we design an adaptive mechanism to modulate the global generator model into local models to handle the differences across entities and relations, which aims to increase the individual informativeness of the samples in a finer-grained manner. In particular, we employ a Feature-wise Linear Modulation (FiLM) layer [23] that conditions the generator on a given entity or entity-relation input.

In summary, we make the following contributions. (1) We design a two-way generator to produce diverse negative triplets, to increase the overall informativeness. (2) We employ a FiLM layer to adapt the global generator model into local models, to increase the individual informativeness of the negative triplets. (3) We conduct extensive experiments on three benchmark datasets. The results demonstrate the superiority of our proposed approach.

2 Background

Negative sampling is an important issue in various machine learning tasks such as recommendation systems [25] and natural language processing [17]. In the context of knowledge graph embedding, negative triplets are often constructed by replacing the tail or head entity in a positive triplet with a randomly sampled entity [2]. Unfortunately, in uniform [2] or popularity-weighted sampling [17], the sampled entity could be completely unrelated to the head or the relation, and therefore be less informative.

To sample more informative negative triplets, researchers have leveraged different heuristics or learning strategies. Structure-aware models [1, 37] exploit the graph structures, which generally select negative examples in the neighborhood of positive examples. For example, SANS [1] hypothesizes that entities that are in close proximity to each other, but do not share a direct relationship, are better candidates for negative sampling. In a similar spirit, PinSage [14] generates localized graphs via random walks to extract informative negative samples. However, these approaches have a high risk of selecting false negatives, as not explicitly related entities in close proximity could still form positive triplets due to the incompleteness of the observed graph.

Other approaches seek to quantify the informativeness of the negative triplets through various learning strategies, including GANs [3, 30], reinforcement learning [31], and importance sampling [39]. These methods provide a more explicit and systematic scoring of negative triplets which often led to better performance. However, these approaches do not consider the diversity and adaptiveness of negative sampling, which are crucial to the overall and individual informativeness of the negative triplets, respectively.

Besides, recent studies [36, 24] show that the optimal negative sampling distribution should be positively but sub-linearly correlated to the positive sampling distribution. Although our proposed model shares a similar view by learning the underlying distribution of positive samples to produce negative samples, we take one step further to consider the diversity and adaptiveness of the negative samples in an adversarial setting. In particular, toward adaptiveness, we borrow the idea from Feature-wise Linear Modulation (FiLM) [23], which was first introduced in the area of visual question answering. Its mechanism includes a learnable feature-wise affine transformation on the hidden neurons of a neural network, conditioned on an arbitrary input. In our context, we employ a FiLM layer to adapt the global generators into local models conditioned on individual input (entity or relation).

3 Methodology

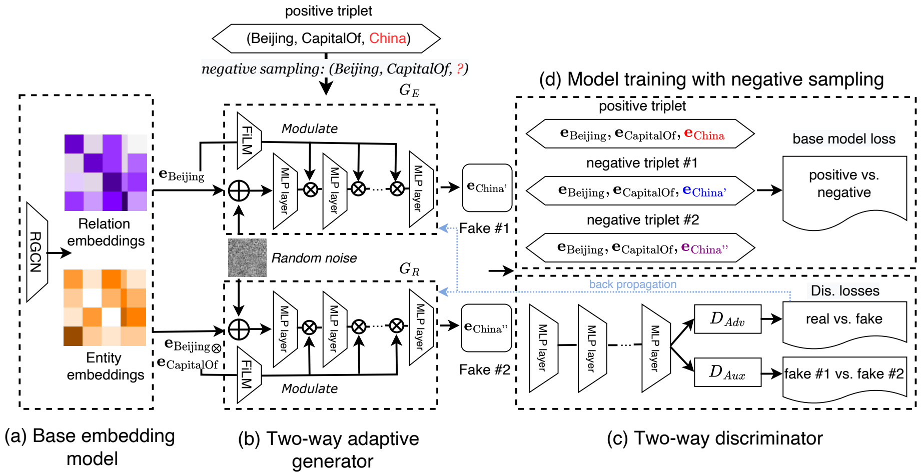
In this section, we introduce the problem formulation and some preliminaries on knowledge graph embedding, followed by our proposed approach DANS.

Before we delve into the details, we first sketch the overall framework in Figure 1. The model consists of four main parts: (a) A *base embedding model* which learns the embeddings for entities and relations; (b) the *two-way adaptive generator* which generate "fake" entity samples to construct negative examples; (c) the *two-way discriminator* which utilize both adversarial and auxiliary losses to improve the quality of produced samples; (d) *model training with negative sampling*, where we replace one entity in a positive triplet with a generated fake entity to form negative triplets, and train the base model together with the original positive triplets.

3.1 Problem formulation and preliminaries

A knowledge graph (KG) is defined by an entity (node) set $\mathcal{V}$, a relation set $\mathcal{R}$ and a ground-truth or positive triplet (edge) set $\mathcal{E}$. Given a triplet $\tau = \langle h, r, t \rangle$ for some $h, t \in \mathcal{V}$ and $r \in \mathcal{R}$, a typical KG model aims to learn a scoring function $\mathcal{F}(\tau)$ to estimate the probability that τ is a positive triplet, *i.e.*, τ is a fact that should appear in the ground truth set $\mathcal{E}$.

Given the power of graph convolutional networks, in this paper, we adopt a multi-layer relational graph convolutional network (RGCN) [26] to serve as our base embedding model in Figure 1(a). The base model encodes the entities in layer $l+1$ into vectors $\mathbf{e}_i^{l+1} \in \mathbb{R}^{d^{l+1}}$ in a latent embedding space, by aggregating


positive triplet
(Beijing, CapitalOf, China)
negative sampling: (Beijing, CapitalOf, ?)
G_E
Modulate
FiLM
MLP layer
$\mathbf{e}_{\text{Beijing}}$
RGCN
Relation embeddings
Entity embeddings
Random noise
G_R
$\mathbf{e}_{\text{Beijing}} \otimes \mathbf{e}_{\text{CapitalOf}}$
$\mathbf{e}_{\text{China'}}$
Fake #1
$\mathbf{e}_{\text{China''}}$
Fake #2
(d) Model training with negative sampling
positive triplet
$\mathbf{e}_{\text{Beijing}}, \mathbf{e}_{\text{CapitalOf}}, \mathbf{e}_{\text{China}}$
negative triplet #1
$\mathbf{e}_{\text{Beijing}}, \mathbf{e}_{\text{CapitalOf}}, \mathbf{e}_{\text{China'}}$
negative triplet #2
$\mathbf{e}_{\text{Beijing}}, \mathbf{e}_{\text{CapitalOf}}, \mathbf{e}_{\text{China''}}$
base model loss
positive vs. negative
back propagation
Dis. losses
D_{Adv}
real vs. fake
D_{Aux}
fake #1 vs. fake #2

(a) Base embedding model

(b) Two-way adaptive generator

(c) Two-way discriminator

Fig. 1: Overall framework of DANS. The toy example only shows how to generate fake tail entities, while generating fake head entities follows a similar process.

their embeddings $\mathbf{e}_j^l \in \mathbb{R}^{d^l}$ from the previous layer l, as follows.

$$\mathbf{e}_i^{l+1} = \text{ReLU}\left(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_i^r} \tfrac{1}{|\mathcal{N}_i^r|} W_r^l \mathbf{e}_j^l + W_0^l \mathbf{e}_i^l\right), \tag{1}$$

where $\mathcal{N}_i^r$ is the set of neighbors of entity i under relation r, W_r^l is a trainable weight matrix for r, W_0^l is an additional trainable weight matrix to capture the self-information of each entity in layer l, and ReLU is the activation function. Assuming a total of L layers are stacked, the embeddings in the last layer are the output embeddings, which we simply write as $\mathbf{e}_i \in \mathbb{R}^d, \forall i \in \mathcal{V}$.

To optimize the parameters, a set of training triplets $\mathcal{D}_{\text{tr}}$ that consists of both positive and negative triplets is used. As shown in Figure 1(d), our objective is to sample a set of high-quality negative triplets, which, together with positive triplets, will be used to minimize the following cross-entropy loss:

$$-\sum_{\tau\in\mathcal{D}_{\text{tr}}} y_\tau \log \mathcal{F}(\tau) + (1-y_\tau)\log(1-\mathcal{F}(\tau)) \tag{2}$$

where $y_\tau = 1$ if $\tau \in \mathcal{E}$, else $y_\tau = 0$. We implement $\mathcal{F}$ using three popular decoders, namely, DistMult [35], ComplEx [29] and RotatE [27]. We provide the DistMult function below, and leave the details of ComplEx and RotatE to Appendix A of the extended version [15].

$$\mathcal{F}(\langle h, r, t\rangle) = \sigma(\mathbf{e}_h^\top \text{Diag}(\mathbf{e}_r)\mathbf{e}_t), \tag{3}$$

where σ is the sigmoid activation, $\mathbf{e}_h, \mathbf{e}_t$ are the head, tail entity embeddings from RGCN, $\text{Diag}(\mathbf{e}_r) \in \mathbb{R}^{d\times d}$ is diagonal matrix whose diagonal is $\mathbf{e}_r$, an r-specific trainable vector of the decoder. Therefore, the full set of training parameters of the base model is $\Theta = \{W_r^l : r \in \mathcal{R}, l \le L\} \cup \{W_0^l : l \le L\} \cup \{\mathbf{e}_r : r \in \mathcal{R}\}$.

3.2 Adaptive two-way generator

A common way to obtain a negative triplet is to replace the tail (or head) entity in a positive triplet by a randomly sampled entity. Beyond simple random sampling, generative adversarial nets (GAN) [10] such as KBGAN [3], IGAN [30], HeGAN [11] and GNDN [38], which learn the underlying sample distributions, have been shown to be effective in negative sampling on KG or other graph structures.

Formally, given a positive triplet $\langle h, r, t\rangle$, a generator G aims to produce a "fake" tail entity t' to replace the real tail t, resulting in a negative triplet $\langle h, r, t'\rangle$. More precisely, G is a function that maps a noise ϵ (typically sampled from a prior distribution) to a vector $\mathbf{e}_{t'}$ in the entity embedding space. Although we follow a similar process, distinct from existing GAN-based approaches, we propose an adaptive two-way generator, as shown in Figure 1(b). It not only diversifies the generation of fake entities, but also localizes the global generator model to adapt to fine-grained differences across entities.

Diversity. Classical GANs generate fake samples through a single pathway and assume a fixed prior distribution, which limits the diversity of fake entity generation. Particularly, in the context of KG, we can generate a fake tail entity associated with either the head entity only, or the relation as well. This improves the diversity of resulting negative triplets and increases the overall informativeness. Hence, we propose a two-way generator that consists of two pathways, namely G_E and G_R, to generate negative triplets associated with a given entity and entity-relation, respectively. Furthermore, having personalized priors for each entity or relation would further enhance the diversification. Specifically, to replace the tail entity in a positive triplet $\langle h, r, t\rangle$ (the same process would also apply to replacing the head entity h), we generate fake tail entity embeddings $\mathbf{e}_{t'}$ and $\mathbf{e}_{t''}$ from the two pathways, as follows.

$$\mathbf{e}_{t'} = G_E(\epsilon; \Theta_{G_E}), \text{ s.t. } \epsilon \sim N(\mathbf{e}_h, \sigma^2 I), \tag{4}$$

$$\mathbf{e}_{t''} = G_R(\epsilon; \Theta_{G_R}), \text{ s.t. } \epsilon \sim N(\mathbf{e}_h \otimes \mathbf{e}_r, \sigma^2 I), \tag{5}$$

where each pathway has its own parameters, *i.e.*, G_E parameterized by Θ_{G_E} and G_R parameterized by Θ_{G_R}. The noise vector ϵ that feeds into each pathway is sampled from a personalized multivariate Gaussian distribution for each entity/relation, $N(\mathbf{e}_h, \sigma^2 I)$ or $N(\mathbf{e}_h \otimes \mathbf{e}_r, \sigma^2 I)$ depending on the pathway. N represents the prior Gaussian distribution for sampling the input to the generator ϵ, σ is a hyper-parameter controlling the covariance of the multivariate Gaussian, I is the identity matrix, and $\otimes$ stands for element-wise multiplication. Intuitively, as the prior Gaussian distributions in Eqs. (4) and (5) are centered on different embeddings, $\mathbf{e}_h$ or $\mathbf{e}_h \otimes \mathbf{e}_r$, it helps to diversify the generated samples from different pathways.

Each pathway is implemented as a multi-layer perceptron (MLP). Taking G_E as an example, its MLP is parameterized by Θ_{G_E} which consists of the weights and biases in each layer. Let $\mathbf{x}^{m+1}_{G_E}$ denote the activations of the m-th MLP layer, where the activations of the last MLP layer are simply the output embedding of G_E. The architecture of G_R mirrors that of G_E.

Adaptiveness. While more diverse samples help increase the overall informativeness, it is also important to improve the informativeness of individual samples. On the one hand, all input entities or relations sharing a global generator model are unable to fully adapt to fine-grained differences across entities or relations. On the other hand, training one model for each entity or relation can cause severe overfitting and incur large overheads. To address the dilemma, we still train a global generator model, but allow the global model to be modulated through a Feature-wise Linear Modulation (FiLM) layer conditioned on each input entity or relation, which essentially adapts the shared global model into local models. Thus, in addition to the global model parameters, the adaptive mechanism only needs to learn the parameters of the FiLM layer, instead of one set of model parameters for each entity or relation.

Consider the pathway G_E to generate a fake tail entity for a head entity h. We adapt the global model G_E to suit the head entity h, by modulating the activations in each hidden layer of G_E:

$$\tilde{\mathbf{x}}^m_{G_E} = \mathbf{x}^m_{G_E} \otimes \alpha^m_h + \beta^m_h, \tag{6}$$

where α^m_h and β^m_h are vectors conditioned on the head entity h and have the same dimension as the m-th layer of G_E. They are used to scale and shift the activations $\mathbf{x}^m_{G_E}$ of the m-th layer of G_E. That is, the global G_E is adapted into a local model conditioned on h. More specifically, α^m_h and β^m_h are output of the FiLM layer F_E applied to the m-th layer of G_E, as follows.

$$\alpha^m_h = F_E(\mathbf{e}_h; \mathit{\Theta}^m_{F_E,\alpha}), \tag{7}$$

$$\beta^m_h = F_E(\mathbf{e}_h; \mathit{\Theta}^m_{F_E,\beta}). \tag{8}$$

Note that the head entity embedding $\mathbf{e}_h$ is the input to F_E, making the output adaptive to and conditioned on h. F_E can be implemented as a MLP, parameterized by $\mathit{\Theta}^m_{F_E,\alpha}$ and $\mathit{\Theta}^m_{F_E,\beta}$ in the m-th layer of G_E. Similarly, the second pathway G_R can be modulated by a FiLM layer F_R, whose input is $\mathbf{e}_h \otimes \mathbf{e}_r$, to generate a fake tail entity for a head entity h and relation r. F_R is parameterized by $\mathit{\Theta}^m_{F_R,\alpha}$ and $\mathit{\Theta}^m_{F_R,\beta}$ in the m-th layer of G_R, to output $\alpha^m_{h,r}$ and $\beta^m_{h,r}$ to scale and shift the activations in G_R.

To sum up, the trainable parameters in the adaptive two-way generator, $\mathit{\Theta}_G$, include the weights in the two global pathways and the FiLM layer weights for each layer in each pathway. For a total of M hidden layers in the global pathways, we have $\mathit{\Theta}_G = \{\mathit{\Theta}_{G_E}, \mathit{\Theta}_{G_R}\} \cup \{\mathit{\Theta}^m_{F_E,\alpha}, \mathit{\Theta}^m_{F_E,\beta}, \mathit{\Theta}^m_{F_R,\alpha}, \mathit{\Theta}^m_{F_R,\beta} : m \leq M\}$.

3.3 Two-way discriminator

As in a standard GAN, a discriminator is needed to help the generator produce high-quality fake entities that mimic real entities. Specifically, the discriminator and the generator compete with each other in a minimax game, in which the generator aims to fool the discriminator by producing realistic looking entities, while the discriminator aims to beat the generator by distinguishing the real and fake entities. In our case, given the two-way generator, we further equip the discriminator with the ability to distinguish the fake entities generated by the two pathways, which can further differentiate and diversify the two pathways.

Concretely, as shown in Figure 1(c), the discriminator also has two pathways: D_{Adv}, an *adversarial* pathway to distinguish fake and real entities, and D_{Aux}, an *auxiliary* pathway to distinguish fake entities generated by G_E and G_R. Taking the generation of tail entities as an example, given the real tail entity t in a positive triplet, as well as the fake entities t' generated by G_E and t'' generated by G_R, D_{Adv} tries to distinguish t from t' and t'', while D_{Aux} tries to distinguish t' from t''. In other words, each of them involves a binary classification:

$$\hat{y}_{\text{Adv},i} = D_{\text{Adv}}(\tilde{\mathbf{e}}_i; \Theta_{D_{\text{Adv}}}), \tag{9}$$

$$\hat{y}_{\text{Aux},i} = D_{\text{Aux}}(\tilde{\mathbf{e}}_i; \Theta_{D_{\text{Aux}}}), \quad \text{s.t. } i \neq t, \tag{10}$$

where D_{Adv} and D_{Aux} are implemented as a fully connected layer, and $\tilde{\mathbf{e}}_i = \textsc{Mlp}(\mathbf{e}_i; \Theta_{D_S})$ is a shared hidden representation computed from the embedding $\mathbf{e}_i$ of a real or fake entity i. The shared hidden representation allows both D_{Adv} and D_{Aux} to benefit from each other during training [20], as they collectively try to distinguish three different classes of samples (t, t', t'').

Note that $\hat{y}_{\text{Adv},i}$ (or $\hat{y}_{\text{Aux},i}$) is the predicted value of the ground-truth label $y_{\text{Adv},i}$ (or $y_{\text{Aux},i}$), such that $y_{\text{Adv},i} = 1$ if i is a real entity, else $y_{\text{Adv},i} = 0$. Furthermore, for a fake entity i, we define $y_{\text{Aux},i} = 1$ if $\mathbf{e}_i$ is generated via Eq. (4), or 0 if generated via Eq. (5). Subsequently, we employ a cross-entropy loss on the two discriminator pathways:

$$\mathcal{L}_{\text{Adv}}(\hat{y}_{\text{Adv},i}, y_{\text{Adv},i}) = -y_{\text{Adv},i} \log \hat{y}_{\text{Adv},i} - (1 - y_{\text{Adv},i}) \log(1 - \hat{y}_{\text{Adv},i}), \tag{11}$$

$$\mathcal{L}_{\text{Aux}}(\hat{y}_{\text{Aux},i}, y_{\text{Aux},i}) = -y_{\text{Aux},i} \log \hat{y}_{\text{Aux},i} - (1 - y_{\text{Aux},i}) \log(1 - \hat{y}_{\text{Aux},i}), \tag{12}$$

In summary, the set of trainable parameters of the two-way discriminator pathway includes the shared parameters and the weights of each classifier, *i.e.*, $\Theta_D = \{\Theta_{D_S}, \Theta_{D_{\text{Adv}}}, \Theta_{D_{\text{Aux}}}\}$.

3.4 Adversarial training

Lastly, we train the generator, discriminator, and base embedding model jointly. On the one hand, the generator aims to fool the adversarial pathway of the discriminator, making D_{Adv} harder to distinguish real and fake entities, as below.

$$\begin{aligned} \arg\max\nolimits_{\Theta_G} \; & \mathbb{E}_t \mathcal{L}_{\text{Adv}}(\hat{y}_{\text{Adv},t}, 1) + \mathbb{E}_{t'} \mathcal{L}_{\text{Adv}}(\hat{y}_{\text{Adv},t'}, 0) + \mathbb{E}_{t''} \mathcal{L}_{\text{Adv}}(\hat{y}_{\text{Adv},t''}, 0) \\ & + \lambda \textstyle\sum_{m,h} (\|\alpha_h^m - \mathbf{1}\|^2 + \|\beta_h^m\|^2), \end{aligned} \tag{13}$$

where t is a real tail entity, and t', t'' are fake tail entities from G_E and G_R, respectively (again, we only illustrate the case where the tail entity in a positive triplet is replaced). The last term in Eq. (13) is a regularization term on the scaling and shifting factors to prevent overfitting [21], and λ is a hyper-parameter to control the strength of regularization. On the other hand, the goal of the discriminator is to overcome the generators by distinguishing fake and real entities, as well as fake entities from different generator pathways, as follows.

$$\arg\min_{\Theta_D}\ \mathbb{E}_i \mathcal{L}_{\text{Adv}}(\hat{y}_{\text{Adv},i}, y_{\text{Adv},i}) + \mathbb{E}_{i \neq t} \mathcal{L}_{\text{Aux}}(\hat{y}_{\text{Aux},i}, y_{\text{Aux},i}), \tag{14}$$

where i can be either real or fake entity in the first term, but $i \neq t$ can only be a fake entity in the second term.

Following a typical adversarial training scheme in negative sampling on knowledge graphs in KBGAN [3], we alternate the model updating among the three parties, as follows. First, we train the generator by updating the generator parameters Θ_G with Eq. (13), while freezing the discriminator parameters Θ_D and the base model parameters Θ. Next, we update Θ_D with Eq. (14), while freezing Θ_G, Θ. Finally, we update Θ by minimizing the loss on the positive and negative triples in Eq. (2), while freezing the other two parameter sets. We repeat the three steps until the convergence of all parties are achieved.

4 Experiments

We perform empirical evaluation on three benchmark knowledge graphs. We first compare the empirical performance of the proposed model DANS[4] with state-of-the-art baselines. In addition, we seek to address a number of research questions (RQs). **RQ1**: Does the two-way design in the generator improve model performance? **RQ2**: Does the adaptive FiLM layer in the generator improve model performance? **RQ3**: What is the impact of the number of negative triplets and adaptive regularization, respectively? **RQ4**: Can we observe the diversity of generated triplets?

4.1 Experimental Design

Datasets. Three benchmark knowledge graphs are used for our experiment. (1) **WN18RR** [2], a harder variant of WN18 [7], which is derived from WordNet consisting of hyponym and hypernym relations between words. Compared to WN18, WN18RR removes inverse relations to minimize leakage from training. (2) **NELL-995** [8] is a subset of the web-based facts collected by the 995th iteration of the Nell system [5] which contains a large pool of entity types and only the top 200 relations are retained. (3) **UMLS** [7] is a specialized knowledge base containing medical entities and their semantic relationships. The entities are biomedical concepts (e.g., disease, antibiotic), and the relations include interactions such as `treats` and `diagnosis`. Table 1 gives a summary of the datasets.

Task and evaluation. We employ the standard knowledge graph completion task [18, 4, 12, 6]. Specifically, for each positive test triplet, we construct a list of candidate triplets that also include negative triplets, which are obtained by

[4] https://github.com/liuran998/DANS

Table 1: Statistics of datasets.

	Entities	Relations	Train	Validation	Test	Total
WN18RR	40,943	11	86,835	3,034	3,134	93,003
NELL-995	63,916	198	137,465	5000	5000	147,465
UMLS	135	46	5,216	652	661	6,529

replacing either the head or tail of the positive triplet with every other entity in the dataset. To avoid false negatives, we follow previous work [2] by adopting their "filtered setting". We then rank the candidate triplets based on the scoring function. For evaluation, we adopt several standard ranking metrics including Mean Reciprocal Ranking (MRR), Hit ratio at 1 (H@1) and Normalized discounted cumulative gain at 5 (NDCG@5) [28]. Details of these ranking metrics can be found in Appendix B of the extended version [15].

Baselines. We compare with baselines in two distinct categories:

(1) *Negative samplers* with the same RGCN backbone [26] and decoders. In other words, they are flexible "plug-ins" that only replace the sampling strategy for a fair comparison to our method DANS. They include **Rand**, which replaces the head or tail entity with a uniformly sampled random entity; **Pop** [17]: a variant of Rand that substitutes uniform sampling with popularity-weighted sampling; **Self-adv** [28]: a self-adversarial negative sampling methodology; **MCNS** [36]: a model which derives negative samples from a distribution that is positively but sub-linearly correlated with the positive distribution.

(2) *Other state-of-the-art baselines* for knowledge graph embedding which may employ a variety of different backbones, heuristics and techniques that diverge from DANS, for a comprehensive comparison. They include **SANS-RW** [1]: a structure-aware model that selects negative samples at close proximity from positive nodes via random walks on the graph; **NSCaching** [39]: a model that employs importance sampling to sample more informative negative triplets; **KBGAN** [3]: a GAN-based model that learns to generate informative negative triplets; **CAKE** [19]: a framework which leverages extra information such as entity types to from factual triplets to sample negative triplets; **SMiLE** [22]: a framework which employs specific contextual information influenced by entity types to sample negative triplets.

Parameter settings. Our model DANS and other negative samplers (Random, Pop, Self-Adv and MCNS) employ RGCN [26] as the backbone, which follows JinheonBaek's pytorch implementation. RGCN is first pre-trained for 15000 epochs, and our base embedding model is then initialized using the pre-trained weights. We train the model for 5000 epochs, using a learning rate of 0.001 and a mini-batch size of 1000 for UMLS, WN18RR and NELL-995. In each mini-batch, the generator and discriminator epochs are set to 5 and 1, respectively, and their learning rates are set to 1e-3 and 1e-4, respectively. The regularization coefficient λ for the FiLM layer in Eq. (13) is set to 1e-4 for all three datasets as it is the most optimal among candidate set {1e-2, 1e-3, 1e-4, 1e-5, 1e-6}.

Furthermore, we generate $N_s = 20$ negative triplets for each positive triplet, out of which the first ten negative triplets are equally split between the two generator pathways, while the remaining ten negative triplets are obtained via uniform random sampling to further increase the diversity. In all cases, either the head or tail of the positive triplets are randomly replaced with negative entities, but not both. We set the output embedding dimension d to 100 for all methods, except SANS-RW, where d is set to the recommended 1,000 to achieve optimal performance. RGCN, RGCN-P, RGCN-Adv and RGCN-MCNS follow the same implementation and settings per the backbone of DANS.

In addition, the hyper-parameters related to negative sampling via Metropolis-Hastings in RGCN-MCNS follow the original paper's link prediction experiments [36]. To reduce the variance resulting from parameter initialization, the experimental results are calculated from an average of five runs with different seeds in all methods. Furthermore, every method is standardized to use the triplet loss in Eqs.(2). Other baseline settings have also been tuned according to the recommendations of the literature. Additional details can be found Appendix C of the extended version [15].

4.2 Results and Analysis

Table 2 reports quantitative comparison against the first category of baselines involving different negative samplers under the same backbone and decoder. Overall, our model DANS consistently leads to better performance for DistMult, RotatE and ComplEx decoders. This shows the robustness of our approach across various decoders. In general, DANS performs better than Rand and its variant Pop, showing that it is important to account for the informativeness of negative triples which are missing in random and popularity-weighted sampling. Since Self-Adv accounts for the informativeness by giving more weight to higher quality triplets, it generally outperforms Rand and Pop. It still lags behind DANS in most cases as it ignores the concepts of diversity and adaptiveness. The variant MCNS shows better performance than Rand and its variant Pop but loses to DANS as MCNS was originally designed for homogeneous graphs.

Next, Table 3 compares DANS with the second category of baselines. Negative sampling in SANS-RW is not relation-aware and thus performs poorly on datasets with more variety of relations, namely, NELL-995 and UMLS. In addition, KBGAN fell short for the two bigger datasets WN18RR and NELL-995 as it ignores graph structure in the sampling process. Furthermore, its adversarial training process potentially suffers from instability and degeneracy. On the other hand, NSCaching employs a more streamlined importance sampling approach, contributing to its competitive performance despite not considering graph structure for negative sampling. As CAKE and SMiLE leverage on extra side information such as entity types to enhance its performances, their experimental results deteriorate as such information are not available in standard knowledge graph completion benchmarks in this paper.

Table 2: Performance comparison with other negative sampling methods, which are plugged into the same backbone (RGCN) and decoders (DistMult, RotatE or ComplEx). The best results are in bold, and the runner-ups are underlined.

Sampling	WN18RR			NELL-995			UMLS		
method	MRR	H@1	NDCG@5	MRR	H@1	NDCG@5	MRR	H@1	NDCG@5
	DistMult								
Rand	$.372_{\pm.002}$	$\underline{.343}_{\pm.003}$	$.369_{\pm.005}$	$.218_{\pm.001}$	$.146_{\pm.002}$	$.219_{\pm.002}$	$.696_{\pm.010}$	$.607_{\pm.082}$	$.693_{\pm.007}$
Pop	$.374_{\pm.002}$	$.342_{\pm.002}$	$\underline{.376}_{\pm.006}$	$.216_{\pm.001}$	$.142_{\pm.002}$	$.216_{\pm.003}$	$.680_{\pm.009}$	$.589_{\pm.012}$	$.692_{\pm.005}$
Self-adv	$.370_{\pm.007}$	$.332_{\pm.010}$	$.373_{\pm.006}$	$\mathbf{.238}_{\pm.003}$	$\underline{.156}_{\pm.003}$	$\mathbf{.241}_{\pm.003}$	$\underline{.717}_{\pm.009}$	$\underline{.624}_{\pm.015}$	$\mathbf{.733}_{\pm.008}$
MCNS	$\underline{.376}_{\pm.004}$	$.340_{\pm.005}$	$.374_{\pm.006}$	$.226_{\pm.002}$	$.144_{\pm.002}$	$.221_{\pm.003}$	$.700_{\pm.002}$	$.606_{\pm.008}$	$.717_{\pm.002}$
DANS	$\mathbf{.381}_{\pm.006}$	$\mathbf{.352}_{\pm.007}$	$\mathbf{.386}_{\pm.008}$	$\underline{.227}_{\pm.004}$	$\mathbf{.162}_{\pm.007}$	$\underline{.220}_{\pm.009}$	$\mathbf{.724}_{\pm.008}$	$\mathbf{.641}_{\pm.009}$	$\underline{.725}_{\pm.008}$
	RotatE								
Rand	$.234_{\pm.009}$	$.110_{\pm.003}$	$.260_{\pm.008}$	$.182_{\pm.003}$	$.093_{\pm.003}$	$\dot{1}89_{\pm.003}$	$.817_{\pm.015}$	$\underline{.683}_{\pm.021}$	$.855_{\pm.013}$
Pop	$.235_{\pm.007}$	$.095_{\pm.003}$	$.268_{\pm.007}$	$.181_{\pm.002}$	$\underline{.131}_{\pm.002}$	$\underline{.200}_{\pm.003}$	$.800_{\pm.005}$	$.673_{\pm.010}$	$.839_{\pm.004}$
Self-adv	$.202_{\pm.007}$	$.058_{\pm.010}$	$.235_{\pm.006}$	$.186_{\pm.002}$	$.096_{\pm.003}$	$.194_{\pm.002}$	$.809_{\pm.007}$	$.677_{\pm.007}$	$.848_{\pm.007}$
MCNS	$\underline{.242}_{\pm.009}$	$\underline{.132}_{\pm.004}$	$\mathbf{.288}_{\pm.006}$	$\underline{.194}_{\pm.003}$	$.122_{\pm.004}$	$\underline{.200}_{\pm.004}$	$\underline{.822}_{\pm.005}$	$.682_{\pm.006}$	$\mathbf{.884}_{\pm.006}$
DANS	$\mathbf{.249}_{\pm.002}$	$\mathbf{.154}_{\pm.001}$	$\underline{.274}_{\pm.003}$	$\mathbf{.195}_{\pm.010}$	$\mathbf{.135}_{\pm.011}$	$\mathbf{.208}_{\pm.010}$	$\mathbf{.833}_{\pm.004}$	$\mathbf{.716}_{\pm.006}$	$\underline{.866}_{\pm.005}$
	ComplEx								
Rand	$.386_{\pm.007}$	$\underline{.346}_{\pm.005}$	$.390_{\pm.006}$	$.245_{\pm.004}$	$.172_{\pm.003}$	$\dot{2}51_{\pm.006}$	$.898_{\pm.008}$	$.822_{\pm.017}$	$.920_{\pm.015}$
Pop	$.389_{\pm.011}$	$.341_{\pm.007}$	$.387_{\pm.012}$	$.241_{\pm.005}$	$.179_{\pm.006}$	$.245_{\pm.004}$	$.840_{\pm.009}$	$.747_{\pm.009}$	$.865_{\pm.008}$
Self-adv	$.375_{\pm.006}$	$.329_{\pm.011}$	$.382_{\pm.013}$	$\underline{.250}_{\pm.005}$	$\underline{.181}_{\pm.007}$	$\mathbf{.277}_{\pm.008}$	$\underline{.908}_{\pm.009}$	$\underline{.844}_{\pm.006}$	$\underline{.925}_{\pm.010}$
MCNS	$\underline{.392}_{\pm.008}$	$.343_{\pm.007}$	$\mathbf{.394}_{\pm.008}$	$.248_{\pm.007}$	$.177_{\pm.004}$	$\underline{.264}_{\pm.009}$	$.879_{\pm.007}$	$.835_{\pm.005}$	$.892_{\pm.011}$
DANS	$\mathbf{.404}_{\pm.005}$	$\mathbf{.347}_{\pm.004}$	$\underline{.392}_{\pm.009}$	$\mathbf{.257}_{\pm.006}$	$\mathbf{.186}_{\pm.010}$	$.255_{\pm.008}$	$\mathbf{.920}_{\pm.007}$	$\mathbf{.857}_{\pm.011}$	$\mathbf{.927}_{\pm.008}$

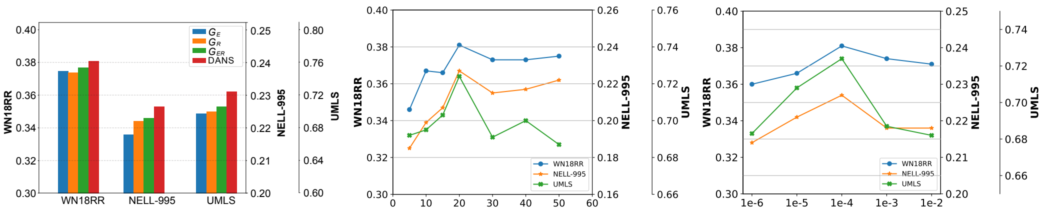
Table 3: Performance comparison with baselines (all using the DistMult decoder). See Table 2 caption for entry styles.

	WN18RR			NELL-995			UMLS		
Model	MRR	H@1	NDCG@5	MRR	H@1	NDCG@5	MRR	H@1	NDCG@5
SANS-RW	$.349_{\pm.010}$	$.340_{\pm.013}$	$.334_{\pm.010}$	$.135_{\pm.006}$	$.109_{\pm.008}$	$.110_{\pm.008}$	$.510_{\pm.008}$	$.369_{\pm.009}$	$.478_{\pm.003}$
NSCaching	$\underline{.374}_{\pm.002}$	$.337_{\pm.003}$	$\underline{.374}_{\pm.002}$	$.177_{\pm.004}$	$\underline{.150}_{\pm.003}$	$.140_{\pm.002}$	$.625_{\pm.004}$	$.508_{\pm.021}$	$.607_{\pm.004}$
KBGAN	$.172_{\pm.004}$	$.070_{\pm.006}$	$.155_{\pm.002}$	$.170_{\pm.002}$	$.077_{\pm.004}$	$\underline{.195}_{\pm.009}$	$\underline{.680}_{\pm.005}$	$\underline{.556}_{\pm.023}$	$\underline{.654}_{\pm.004}$
CAKE	$.353_{\pm.007}$	$\underline{.345}_{\pm.005}$	$.351_{\pm.008}$	$\underline{.204}_{\pm.006}$	$.130_{\pm.007}$	$.175_{\pm.012}$	$.441_{\pm.013}$	$.365_{\pm.008}$	$.383_{\pm.010}$
SMiLE	$.315_{\pm.006}$	$.291_{\pm.007}$	$.294_{\pm.012}$	$.131_{\pm.004}$	$.127_{\pm.005}$	$.105_{\pm.008}$	$.414_{\pm.015}$	$.345_{\pm.007}$	$.372_{\pm.013}$
DANS	$\mathbf{.381}_{\pm.006}$	$\mathbf{.352}_{\pm.007}$	$\mathbf{.386}_{\pm.008}$	$\mathbf{.227}_{\pm.004}$	$\mathbf{.162}_{\pm.007}$	$\mathbf{.220}_{\pm.009}$	$\mathbf{.724}_{\pm.008}$	$\mathbf{.641}_{\pm.009}$	$\mathbf{.725}_{\pm.008}$

Overall, DANS has obtained favourable performance, showing the importance of diversity and adaptiveness during negative sampling. We will conduct further ablation study in the next part to examine the contribution from each aspect. Finally, we have included the experimental results for dataset FB15k-237 which show favorable performance on ComplEx decoder in Appendix D of the extended version [15].

4.3 Additional research questions

In this part, we seek to investigate RQ1–RQ4 listed at the beginning of this section. All experiments in this part are conducted using the DistMult function as the decoder.



(a) Ablation study (b) No. of negative triplets N_s (c) Extent of regularization λ

Fig. 2: Investigation of research questions. Each dataset uses its own y-axis, and the metric used on the y-axes is MRR. (a) Ablation study on the contribution of individual generator pathways and adaptive FiLM layer. (b) Study of parameter sensitivity on the number of negative triplets N_s and (c) extent of adaptive regularization λ.

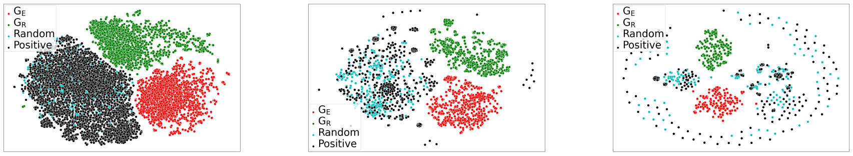
Ablation study (RQ1, RQ2). We investigate the contribution from major design choices through an ablation study. As depicted in Figure 2(a), we compare DANS with the following variants, all of which *do not employ the FiLM layer*. (1) G_E: Only the pathway G_E in the generator; (2) G_R: Only the pathway G_R in the generator; (3) G_{ER}: Both pathways G_E and G_R.

From the results, among the single pathways (either G_E or G_R), there is no consistent winner and it depends on the dataset. However, it is clear that the use of both pathways in the generator (G_{ER}) outperforms using just a single pathway. Thus, this addresses RQ1 and shows that diversifying the negative triplets with the two-way generator can improve model performance and improve the overall informativeness of the negative triplets.

Furthermore, by comparing G_{ER} (*i.e.*, both pathways without FiLM) and the proposed model DANS (*i.e.*, both pathways with FiLM), our model obtains a significant lead in performance. This addresses RQ2 and shows the effectiveness of our adaptive design using FiLM.

Parameter sensitivity (RQ3). To answer RQ3, we perform a parameter sensitivity analysis. We first analyze how the number of negative triplets per positive triplet, N_s, can impact model performance. As shown in Figure 2(b), as we increase N_s on each dataset, we consistently observe that the MRR performance improves and peaks at $N_s = 20$. A larger N_s allows for greater diversity, which explains the initial increase in performance. However, when $N_s \geq 20$, performance starts to plateau or even deteriorate, due to imbalanced training data.

Next, we investigate the impact of adaptive regularization controlled by λ in Figure 2(c). Generally, having such a regularization (*i.e.*, $\lambda > 0$) avoids excessive scaling and shifting from the FiLM layer, and thus reduces overfitting to individual entities or relations. In particular, the MRR performance improves as λ increases and achieves the most optimal performance for all three datasets when λ is around 1e-4. As the optimal values of N_s and λ are largely stable across the three datasets, our model is not sensitive to these hyperparameter settings,



(a) `hasPart` (WN18RR) (b) `animalType` (NELL-995) (c) `isA` (UMLS)

Fig. 3: Visualization of diversity. Best viewed in color.

and potentially requires less effort in hyperparameter tuning. We also note that the performance on the UMLS datasets tends to be more sensitive to changes in both parameters. This could be because UMLS is a smaller dataset than the other two, containing only 5,216 positive triplets in training and this increases the risk of overfitting to certain settings in general.

Case studies (RQ4). We conduct a qualitative evaluation of DANS to demonstrate the diversity of negative triplets generated by DANS. While the ablation study has demonstrated the importance of diversity through improved model performance with the two-way generator, we further present a few case studies of how DANS can produce more diverse examples than uniform random sampling (RNS). In Figures 3, we visualize the positive and negative tail entities w.r.t. a given relation and all its head entities on each dataset. More specifically, each point represents one tail entity, which can be a positive (real) tail entity, or a negative tail entity. The negative entity can be generated by one of the pathways G_E or G_R of the generator, or randomly sampled by RNS. The high-dimensional embedding space is projected onto a Cartesian plane using the *t*-SNE algorithm [16]. In Figure 3, we compare the diversity of negative entities generated by DANS with that of RNS-based negative entities. For our case study, we select one relation for each dataset, namely, `hasPart` on WN18RR, `animalType` on NELL-995 and `isA` on UMLS, so that all the positive and negative tail entities for a common relation (and the same original head entities) can be contrasted in one visualization. The results show that DANS could provide more diverse negative entities for model training, where those generated by G_E and G_R occupy different subspaces from the positive entities. In contrast, RNS lacks diversity and samples negative entities in the same subspace as positive entities. This could even potentially contribute to false negative triplets as they are not well separated from the real ones.

5 Conclusion

In this work, we introduced DANS, a negative sampling strategy for knowledge graph embedding that explicitly accounts for the informativeness of negative triplets. On one hand, we proposed a two-way generator to increase the overall informativeness by diversifying the negative triplets based on their association with not only entities but also relations. On the other hand, we adapt the global generator model into local models, which generate negative triplets in a finer-grained manner to improve their individual informativeness. Empirically, DANS has outperformed state-of-the-art baselines on three benchmark knowledge graphs through both quantitative and qualitative experiments.

References

1. Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W.L., Bose, A.J.: Structure aware negative sampling in knowledge graphs. arXiv preprint arXiv:2009.11355 (2020)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. NeurIPS **26** (2013)
3. Cai, L., Wang, W.Y.: Kbgan: Adversarial learning for knowledge graph embeddings. arXiv preprint arXiv:1711.04071 (2017)
4. Cao, Y., Wang, X., He, X., Hu, Z., Chua, T.S.: Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In: WWW. pp. 151–161 (2019)
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E., Mitchell, T.: Toward an architecture for never-ending language learning. In: AAAI. pp. 1306–1313 (2010)
6. Chen, Z., Wang, Y., Zhao, B., Cheng, J., Zhao, X., Duan, Z.: Knowledge graph completion: A review. Ieee Access **8**, 192435–192456 (2020)
7. Dettmers, T., Pasquale, M., Pontus, S., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: AAAI (2018)
8. Dutta, S., Gittens, A., Zaki, M.J., Aggarwal, C.C.: Replacing paths with connection-biased attention for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 14885–14892 (2025)
9. Fang, Y., Kuan, K., Lin, J., Tan, C., Chandrasekhar, V.: Object detection meets knowledge graphs. In: IJCAI (2017)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. NeurIPS **27** (2014)
11. Hu, B., Fang, Y., Shi, C.: Adversarial learning on heterogeneous information networks. In: KDD. pp. 120–129 (2019)
12. Ji, G., Liu, K., He, S., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: AAAI (2016)
13. Li, S., Wong, K.W., Fung, C.C., Zhu, D.: Improving question answering over knowledge graphs using graph summarization. In: ICONIP. pp. 489–500. Springer (2021)
14. Liu, H., Hu, K., Wang, F.L., Hao, T.: Aggregating neighborhood information for negative sampling for knowledge graph embedding. NCA **32**, 17637–17653 (2020)
15. Liu, R., Liu, Z., Li, X., Wu, H., Fang, Y.: Diversified and adaptive negative sampling on knowledge graphs (2024), https://arxiv.org/abs/2410.07592
16. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. JMLR **9**(11) (2008)
17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. NeurIPS **26** (2013)
18. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. arXiv preprint arXiv:1906.01195 (2019)

19. Niu, G., Li, B., Zhang, Y., Pu, S.: Cake: A scalable commonsense-aware framework for multi-view knowledge graph completion. arXiv preprint arXiv:2202.13785 (2022)
20. Odena, A.: Semi-supervised learning with generative adversarial networks. arXiv preprint arXiv:1606.01583 (2016)
21. Oreshkin, B., Rodríguez López, P., Lacoste, A.: Tadam: Task dependent adaptive metric for improved few-shot learning. NeurIPS **31** (2018)
22. Peng, M., Liu, B., Xie, Q., Xu, W., Wang, H., Peng, M.: Smile: Schema-augmented multi-level contrastive learning for knowledge graph link prediction. arXiv preprint arXiv:2210.04870 (2022)
23. Perez, E., Strub, F., De Vries, H., Dumoulin, V., Courville, A.: Film: Visual reasoning with a general conditioning layer. In: Proceedings of the AAAI conference on artificial intelligence (2018)
24. Qian, J.: Understanding negative sampling in knowledge graph embedding. IJAIA **12**(1) (2021)
25. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
26. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607. Springer (2018)
27. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197 (2019)
28. Sun, Z., Yu, D., Fang, H., Yang, J., Qu, X., Zhang, J., Geng, C.: Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In: RecSys. pp. 23–32 (2020)
29. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International conference on machine learning. pp. 2071–2080. PMLR (2016)
30. Wang, P., Li, S., Pan, R.: Incorporating gan for negative sampling in knowledge representation learning. In: AAAI (2018)
31. Wang, X., Xu, Y., He, X., Cao, Y., Wang, M., Chua, T.S.: Reinforced negative sampling over knowledge graph for recommendation. In: WWW. pp. 99–109 (2020)
32. Wu, Q., Wang, P., Shen, C., Dick, A., Van Den Hengel, A.: Ask me anything: Free-form visual question answering based on knowledge from external sources. In: CVPR. pp. 4622–4630 (2016)
33. Xi, W.D., Huang, L., Wang, C.D., Zheng, Y.Y., Lai, J.H.: Deep rating and review neural network for item recommendation. TNNLS **33**(11), 6726–6736 (2021)
34. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: A reinforcement learning method for knowledge graph reasoning. arXiv preprint arXiv:1707.06690 (2017)
35. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
36. Yang, Z., Ding, M., Zhou, C., Yang, H., Zhou, J., Tang, J.: Understanding negative sampling in graph representation learning. In: KDD. pp. 1666–1676 (2020)
37. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: KDD. pp. 974–983 (2018)
38. Zeng, J., Liu, L., Zheng, X.: Learning structured embeddings of knowledge graphs with adversarial learning framework. arXiv preprint arXiv:2004.07265 (2020)
39. Zhang, Y., Yao, Q., Shao, Y., Chen, L.: Nscaching: simple and efficient negative sampling for knowledge graph embedding. In: ICDE. pp. 614–625. IEEE (2019)