# NoiGAN: Noise Aware Knowledge Graph Embedding with Adversarial Learning

Kewei Cheng
Department of Computer Science, University of California, Los Angeles
viviancheng@cs.ucla.edu

Yikai Zhu
Department of Computer Science, Peking University
zyksir@outlook.com

Ming Zhang
Department of Computer Science, Peking University
mzhang@net.pku.edu.cnt

Yizhou Sun
Department of Computer Science, University of California, Los Angeles
yzsun@cs.ucla.edu

## ABSTRACT

Although knowledge graphs (KGs) have gained increasing attention in recent years for their successful applications in numerous applications across different domains, the extensive power of KGs is severely limited by the incompleteness of the real-world data and the inevitably involvement of various kinds of errors. On one hand, although knowledge graph embedding (KGE) methods show strong reasoning ability in inferring missing facts, they usually lack the capability to handle noise. On the other hand, even though several attempts have been made to automatically remove noisy triples from KGs, they treat error detection as an orthogonal task to KGE methods. However, intuitively, embedding and error detection tasks are inter-dependent and can mutually enhance each other. Error detection is extremely useful to prepare clean KG for KGE model while high quality embedding learned by KGE model is powerful for identifying noisy data. In this paper, we propose a unified adversarial learning-based model, called NoiGAN, to jointly train these two tasks. Extensive experiments have demonstrated that our approach is superior to existing state-of-the-art algorithms in regard to both KG completion and error detection.

**ACM Reference Format:**
Kewei Cheng, Yikai Zhu, Ming Zhang, and Yizhou Sun. 2020. NoiGAN: Noise Aware Knowledge Graph Embedding with Adversarial Learning. In *Proceedings of The Second International Workshop on Deep Learning on Graphs: Methods and Applications (DLG-KDD'20).* , 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Although knowledge graphs (KGs) have gained increasing attention in recent years for their successful applications in numerous applications across different domains, such as web search [7] and question answering [9, 29], the extensive power of KGs is severely limited by the incompleteness of the real-world data and the inevitably involvement of various kinds of errors [17].

Knowledge graph embedding-based (KGE) methods currently hold the state-of-the-art in KG completion for their strong reasoning ability and good scalability [1, 11, 13, 20, 21, 25, 28]. However, they usually lack the ability to handle noisy data. In practice, the automatic construction process of KGs inevitably introduces many kinds of errors, including conflicting, erroneous, and ambiguous information [17]. To remove the noise from KGs, various KG error detection methods can be utilized to provide clean KGs for KGE models by identifying and remove incorrect triples in the KGs automatically [5, 26]. The main challenge for KG error detection is the scarcity of labeled noisy triples since samples of noisy triples is usually costly and labor intensive to obtain in real-world. As a result, most of existing methods [17] utilize the ontology information and human defined logical rules to detect error in KGs. They measure the correctness of a triple by determining whether it violates the ontology structure or logical rules.

Although existing KG error detection methods treat itself as orthogonal task to KGE methods, intuitively, KG embedding methods could be potentially useful for KG error detection task. Considering the strong power of KGE in capturing network structure, high-quality embeddings could reveal global structure information of KGs. Despite the difficulty in collecting noisy triples, as long as KGE provides deep understanding of network structure of KGs, we can generate the noise instead. In addition, recent studies on the memorization effects of deep neural networks show that neural networks would first memorize training data of clean labels and then those of noisy labels [8]. Thus, KGE is also useful to select credible correct triples from noisy KGs. With both selected reliable true triples and generated noisy triples, KG error detection can be easily achieved by training a classifier to distinguish noisy triples from the correct triples.

Despite the potential benefits brought by the mutual interaction between KGE and error detection, a vast majority of existing works fail to capture it by treating KGE and KG error detection as orthogonal tasks. To address the limitation of existing works, in this paper, we propose a novel framework NoiGAN to jointly combine KGE and error detection with a unified generative adversarial network (GAN) [6]. In general, NoiGAN consists of two components: (1) a noise-aware KGE module to learn robust representations of entities and relations, and (2) an adversarial learning module to distinguish


*DLG-KDD'20, August 24, 2020, Virtual Conference*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

noise from normal data. During the training, the noise-aware KGE model takes the confidence score computed by GAN, which indicate the degree of trustworthiness of a triple fact, as the guidance to eliminate the noisy data from the training data whereas the GAN requires the noise-aware KGE model to continuously provide high quality embedding as well as credible correct triples to train a classifier which can distinguish noise from normal data. Cooperation between these two components drives both to improve their capability. The main contributions of this paper are summarized as follows:

- We propose a unified framework, known as NoiGAN, to leverage the mutual interaction between KGE and error detection for noise-aware KG embedding learning. Under the framework, KGE and error detection could alternately and iteratively boost performance of each other.
- Our proposed framework can be easily generalized to various KGE models to enhance their ability in dealing with noisy knowledge graph.
- We experimentally demonstrate that our new algorithm is superior to existing state-of-the-art algorithms in terms of both knowledge graph completion and noise detection.

## 2 PROBLEM STATEMENT

In this section, we formally define our problem in order to clarify the differences between our setting and previous work. Notation wise, we use bold lowercase letters to denote vectors (e.g., $\mathbf{h}$), and regular lowercase letters to denote single variables (e.g., $h$).

Given a knowledge graph $\mathcal{G} = \{E, R, \mathcal{T}\}$, which consists of a set of entities $E$, a set of relations $R$ and a set of observed facts $\mathcal{T} = \{(h, r, t) \mid h, t \in E,\ r \in R\}$, where a fact is represented by a triple $(h, r, t)$, and $h$, $t$, and $r$ denotes its head entity, tail entity, and relation, respectively.

Let $\mathcal{G} = \{E, R, \mathcal{T}\}$ be a knowledge graph, which consists of a set of entities $E$, a set of relations $R$ and a set of observed facts $\mathcal{T} = \{(h, r, t) \mid h, t \in E,\ r \in R\}$, where a fact is represented by a triple $(h, r, t)$, and $h$, $t$, and $r$ denotes its head entity, tail entity, and relation, respectively. Their corresponding embedding vectors are denoted as $\mathbf{h}$, $\mathbf{t}$ and $\mathbf{r}$ accordingly. Existing KGE models assume that every triple $(h, r, t) \in \mathcal{T}$ is correct, which is not true in real world. To take the potential noise in KGs into consideration, we divide the observed triples into two disjoint sets according to their correctness, $\mathcal{T}_E$ and $\mathcal{T}_C$, where $\mathcal{T}_E$ denotes the collection of noise while $\mathcal{T}_C$ denotes the collection of correct triples. Note that the partition of $\mathcal{T}_E$ and $\mathcal{T}_C$ is unknown for us. To predict the degree of trustworthiness of a triple, each triple $(h, r, t)$ is associated with a **confidence score** $C(h, r, t)$, which can be either a binary value 0 or 1, or a soft value from $[0, 1]$, indicating how likely the corresponding triple is true. Then, the problem of learning noise aware KG embedding can be stated as follows:

**Given:** A noisy knowledge graph $\mathcal{G}$, including both correct triples $\mathcal{T}_C$ and noise $\mathcal{T}_E$.

**Learn:** the confidence scores $\{C(h, r, t) | \forall (h, r, t) \in \mathcal{T}\}$ as well as a corresponding noise aware KGE model under the guidance of $\{C(h, r, t) | \forall (h, r, t) \in \mathcal{T}\}$.

## 3 THE PROPOSED METHOD: NOIGAN

In this section, we illustrate our proposed framework, NoiGAN, in details. NoiGAN aims to learn noise-aware KG embeddings by jointly training both embedding and error detection tasks under a unified adversarial learning-based framework. It consists of two components: (i) Noise aware KGE model, which incorporates confidence scores (i.e., $C(h, r, t)$) into the KGE model to isolate the impact of noise over embedding vectors. Meanwhile, it provides high quality embeddings as well as reliable correct triples (i.e., $(h, r, t) \in \mathcal{T}_C$) to model GAN (Section 3.1); (ii) Adversarial learning framework (GAN) for noise identification, which can be further divided into two components: (a) Noise generator (i.e., $G((h', r, t') | (h, r, t); \theta_G)$), which learns to generate the most likely triples to be the noise by corrupting the correct triple. It aims at providing reliable noise samples for the discriminator as well as high quality negative samples for noise aware KGE model (Section 3.2.1); (b) Discriminator (i.e.,$D((h, r, t); \theta_D)$), which aims to learn confidence score for each observed triple to distinguish correct triples from noise (i.e.,$\{C(h, r, t) | \forall (h, r, t) \in \mathcal{T}\}$). The confidence score is utilized by the noise aware KGE model to eliminate the noise from the learning process (Section 3.2.2).

### 3.1 Noise Aware Knowledge Graph Embedding Model

Most of existing embedding models assume that all observed triples are completely true in KGs, which is inappropriate. In fact, KGs contains many kinds of errors due to the automatic construction process. To isolate the impact of noise over embedding vectors, following [19], we adopted confidence score to describe whether a triple fact is noisy. In particular, confidence score can be learned by a discriminator, which is trained to correctly classify the noise and the correct triples. We will discuss how to obtain such discriminator in Section 3.2.2. With the learned confidence score, the objective function of noise-aware KGE model becomes as follows, which can be easily adapted to any KGE models by define $f_r(h, t)$ using different scoring functions.

$$\mathcal{L}_{KGE} = \sum_{(h,r,t) \in \mathcal{T}} C(h, r, t).[(-\log \sigma(\gamma - f_r(h, t)) + \sum_{(h',r,t') \in \mathcal{N}(h,r,t)} \frac{1}{|\mathcal{N}(h, r, t)|} \log \sigma(\gamma - f_r(h', t')))]$$

where $\gamma$ is the margin, $\sigma$ is the sigmoid function and $C(h, r, t)$ is the confidence score assigned to each observed triple $(h, r, t) \in \mathcal{T}$. To be specific, $C(h, r, t)$ can be both a binary variable or a soft value from the interval $[0, 1]$. The closer the value is to 0, the greater the probability that the triple is noise. We denote the previous one as the hard version while the later one as the soft version. To reduce the effects of randomness, we sample multiple negative triples $(h', r, t')$ for each observed triple $(h, r, t) \in \mathcal{T}$. We denote the negative triples set of a triple $(h, r, t)$ as $\mathcal{N}(h, r, t)$. Negative triples set is constructed by replacing the head or tail entity of the observed triples with an entity sampled randomly from entity set $E$:

$$\mathcal{N}(h, r, t) = \{(h', r, t) | h' \in E\} \cup \{(h, r, t') | t' \in E\}, (h, r, t) \in \mathcal{T}$$

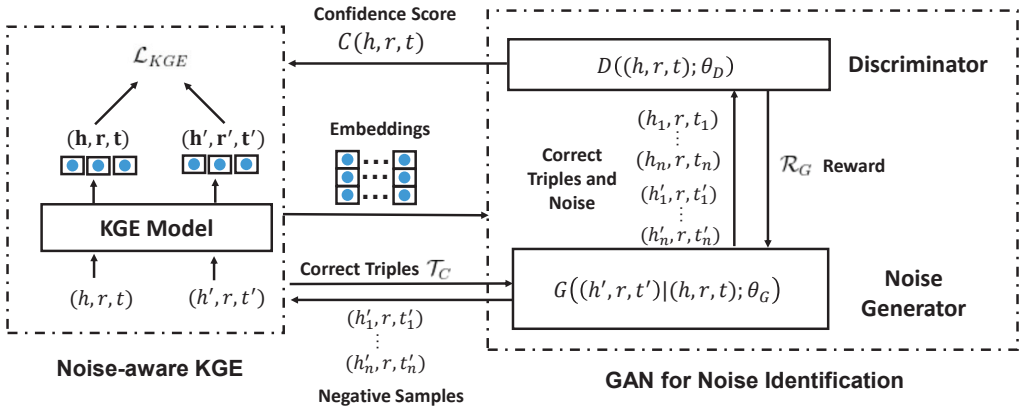



**Figure 1: Illustration of the proposed NoiGAN framework. It consists of two main components, a noise-aware KGE model for embedding learning and a GAN for noise identification.**

## 3.2 GAN for Noise Identification

Our adversarial learning framework for noise identification consists of two components, a noise generator and a discriminator. Unlike traditional GAN whose ultimate goal is to train a good generator, our proposed GAN aims to produce a good discriminator to distinguish noise from true triples. The classification probability predicted by the discriminator will be adopted as the confidence score by noise-aware KGE in Section 3.1. Considering that samples of noise is usually costly and labor intensive to obtain in reality, we thus learn a noise generator to generate the noise instead. In particular, the generator and the discriminator act as two players in a minimax game: the generator learns to continually generate "more confusing" triples and thus provides better quality noise for the discriminator, whereas the discriminator is trained to draw a clear distinction between the correct triples and the noise generated by its opponent generator. Formally, it can be formulated as:

$$\min_G \max_D V(G,D) = \sum_{(h,r,t)\in\mathcal{T}_C} \mathbb{E}_{(h,r,t)\sim\mathcal{T}_C}[\log D((h,r,t);\theta_D)] + \mathbb{E}_{(h',r,t')\sim G(\cdot|(h,r,t);\theta_G)}[\log(1-D((h',r,t');\theta_D))]$$

where $\mathcal{T}_C$ represents the collection of correct triples whose construction will be discussed in Section 3.2.2.

*3.2.1 Noise Generator.* The main goal of the generator is to generator high-quality fake triples that can fool discriminators, which in return, brings a better classifier to identify error in KGs. In addition, to enable KGE model learn from the noisy data simultaneously, the noise generated by the generator can also be taken as negative samples to train the noise-aware KGE model. Specifically, the generator takes embedding vectors as well as the collection of correct triples $\mathcal{T}_C$ learned by the noise-aware KGE model as input. Given a correct triples $(h,r,t) \in \mathcal{T}_C$, the generator aims to select the most likely triples to be the noisy triples $(h',r,t')$ from its negative samples candidate set $\mathcal{N}(h,r,t)$. To achieve this goal, a two-layer fully-connected neural network is introduced to model the probability distribution over the candidate negative samples $\mathcal{N}(h,r,t)$. In particular, this MLP uses ReLU as the activation function for the first layer and adds Softmax to the second layer. It takes the concatenation of the embedding vectors $\mathbf{h}'$, $\mathbf{r}$ and $\mathbf{t}'$ of triple $(h',r,t')$ as input and output the probability whether the triple $(h',r,t')$ is the most likely noisy triple. As the output of the generator is a discrete triple, training for the generator is a little tricky due to data indifferentiability issue. A common solution is to use policy gradient based reinforcement learning instead [22]. In particular, for a generated triple, the reward from discriminator is defined as $f_D(h',r,t')$, which is exactly the probability of the triple $(h',r,t')$ to be true. Thus, in order to fool the discriminator, the generator is trained to maximize the expected reward as follows:

$$\mathcal{R}_G = \sum_{h,r,t} \mathbb{E}_{(h',r,t')\sim G(\cdot|(h,r,t);\theta_G)}[\log f_D(h',r,t')] \tag{1}$$

*3.2.2 Discriminator.* Discriminator is essentially a noisy triple classifier, which aims to distinguish the noise from the true triples. Both correct and noisy triples are required to train the discriminator. Generator can provide reliable noisy triples for the discriminator as introduced in Section 3.2.1. To prepare reliable correct triples, regarding all observed triples in a KG as candidates obviously is inappropriate. Instead, we leverage memorization effects of deep neural network to select qualified candidates from $\mathcal{T}$. According to memorization effects, when noisy data exist, deep learning models tend to memorize these noise in the end and thus leads to the lower prediction score of noisy triples [8]. We therefore take top $\alpha\%$ triples which fit the KGE the best (with lowest $f_r(h,t)$) to construct the correct triples set $\mathcal{T}_C$. Considering discriminator is essentially a binary classifier, the objective function of the discriminator can be formulated as minimizing the following cross entropy loss:

$$\mathcal{L}_D = -\sum_{(h,r,t)\in\mathcal{T}_C} \log(f_D(h,r,t)) - \sum_{(h',r,t')\in G(\cdot|(h,r,t);\theta_G)} \log(1-f_D(h',r,t')) \tag{2}$$

where $f_D(h,r,t) = \sigma(\text{MLP}(h,r,t))$ where MLP is a two-layer neural network with ReLU as activation function and $\sigma(x) = 1/(1+\exp(-x))$ is the standard sigmoid function. If we use TransE as our KGE model, the MLP takes the vector $\mathbf{h}+\mathbf{r}-\mathbf{t}$ as input and output the probability of the triple $(h,r,t)$ being true. $f_D(h,r,t)$ is used to define $C(h,r,t)$ for each observed triple in a KG (i.e., $\{C(h,r,t)|\forall(h,r,t) \in \mathcal{T}\}$). To be specific, $C(h,r,t)$ can be either a binary value or a soft value. When it is a binary variable, it represents the classification result regard to whether a triple is a true triple. When it takes a soft value, it indicates the probability of the triple $(h,r,t)$ being true.

**Algorithm 1** NoiGAN Framework

**Require:** Observed triples in a knowledge graph $\mathcal{T}$
1: Initialize $\{C(h,r,t) = 1|\forall(h,r,t) \in \mathcal{T}\}$.
2: Train noise aware KGE model with random $\mathcal{N}(h,r,t)$.
3: Update embeddings $\mathbf{h}, \mathbf{r}, \mathbf{t}$.
4: **for** $n = 1 : N$ **do**
5: Take top $\alpha\%$ triples with lowest $f_r(h,t)$ as $\mathcal{T}_C$.
6: Train GAN with $\mathcal{T}_C$ and $\mathbf{h}, \mathbf{r}, \mathbf{t}$.
7: Update $\{C(h,r,t)|\forall(h,r,t) \in \mathcal{T}\}$ according to Eq. (2) .
8: Train noise aware KGE model with Update $\{C(h,r,t)|\forall(h,r,t) \in \mathcal{T}\}$ and $\mathcal{N}(h,r,t)$ generated by generator $G$.
9: **end for**

## 3.3 Joint Training

We train each component of NoiGAN alternately by fixing the parameters of the others. We start by training the noise aware KGE model. We initiate the confidence score $C(h,r,t)$ as 1 for all $(h,r,t) \in \mathcal{T}$ at the very beginning. Top $\alpha\%$ triples which fit the noise aware KGE the best will be selected to construct $\mathcal{T}_C$. Then, with both learned embedding vectors $\mathbf{h}, \mathbf{r}, \mathbf{t}$ and $\mathcal{T}_C$, we start the training of the GAN. Since the generator aims to generate the most likely noisy triples $(h', r, t')$ to fool the discriminator, the reward $\mathcal{R}_G$ given by the discriminator is taken as guidance for its training. Meanwhile, the discriminator learns a classifier to distinguish noisy triples by taking triples generated by the generator as negative examples and triples in $\mathcal{T}_C$ as positive examples. Once the training of GAN has done, the discriminator is utilized to update the confidence score $C(h,r,t)$ for all observed triples in the KG. After that, the training of the noise aware KGE model will start again. The process repeats until convergence. The embeddings learned by the noise aware KGE model is regarded as the final representation for entities and relations.

# 4 EXPERIMENTS

## 4.1 Experimental Settings

**Datasets.** We evaluate our NoiGAN on three widely used benchmark datasets, including FB15K-237, WN18RR and YAGO3-10. To simulate the real-world noisy knowledge graphs, we modified these benchmark datasets to include noise. Complete details of injecting noise to benchmark datasets as well as statistics of the datasets are summarized in appendix.

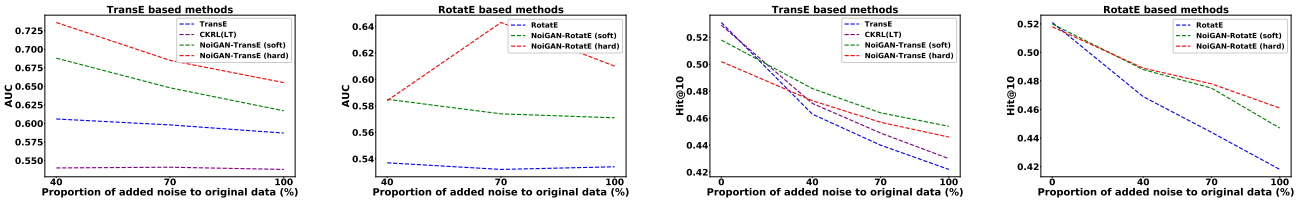
**Baselines.** NoiGAN is compared with the following state-of-the-art algorithm, including (1) KGE models (e.g., TransE [1],DistMult [28] and RotatE [20]); (2) robust KGE models ( e.g., attention based method [15]); (3) KGE models with GAN (e.g., KBGAN [2]) and (4) noise aware KGE models (e.g., CKRL [27]). As there are three kinds of triple confidences defined in CKRL, we take CKRL with local triple confidence, called CKRL (LT), as our baseline. To fairly compare different methods, the same loss function and negative sampling strategies are employed for all models.

**Experimental Setup of NoiGAN.** We evaluate two versions of our NoiGAN model as mentioned in Section 3.2.2. The soft version is denoted as NoiGAN (soft) while the hard version is denoted as NoiGAN (hard). In particular, both TransE [1] and RotatE [20] are implemented as score function for NoiGAN, which are denoted as NoiGAN-TransE and NoiGAN-RotatE respectively. The detailed hyperparameters settings are available in the appendix.

## 4.2 Results

**Knowledge Graph Noise Detection.** To verify the capability of NoiGAN in distinguishing noises in knowledge graphs, we evaluate NoiGAN in terms of classification performance on the training set. The experiments are conducted on three benchmark datasets with injected noisy triples to be the ratio of 40% of true triples. Two classification evaluation metrics are adopted, including (1) AUC; (2) Specificity (True Negative Rate), which can be calculated as $\frac{\text{TN}}{\text{TN+FP}}$, where TN represents true negative while FP represents false positive.The results can be found in Table 1. We have several interesting observations from the results: (1) NoiGAN-TransE (hard) consistently achieves the state-of-the-art performances over all datasets in almost all cases. (2) Both NoiGAN-TransE and NoiGAN-RotatE consistently outperform their base model (i.e., TransE and RotatE) on all datasets with significant performance gain, which demonstrates the capability of NoiGAN framework in detecting noises.

**Knowledge Graph Completion.** To verify the quality of learned embedding, we evaluate NoiGAN over KG completion task to show its reasoning ability. We conduct experiments on three benchmark datasets with noisy triples to be the ratio of 70% of true triples. During evaluation, we mask the head or tail entity of each test triple to predict the masked entity. The Hit@K (H@K) and Mean Reciprocal Rank (MRR) are adopted as the evaluation metrics. Results of reasoning are shown in Table 2. In particular, as attention based method exceed the memory capacity of our machines on YAGO3-10 dataset, only the results on the FB15K-237 and WN18RR datasets are reported. Note that TransE, CKRL (LT) and NoiGAN-TransE share the same score function, they are comparable with each other while RotatE and NoiGAN-RotatE share the same score function RotatE and thus they are comparable with each other. We make the following observations: (1) NoiGAN consistently outperforms the baseline methods which share the same score function with it on noisy dataset sets. The performance gain is significant especially on datasets with 100% noise; (2) Either NoiGAN-TransE or NoiGAN-RotatE will achieve the best performance on almost all noisy datasets. In particular, hard version of NoiGAN performs better then soft version in most cases; (3) Although attention based




(a) KG Noise Detection.

(b) KG Completion.

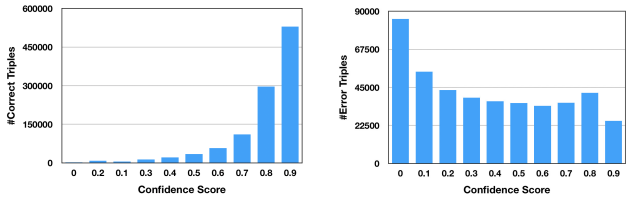
**Figure 2: Robustness Analysis of NoiGAN on FB15K-237 Dataset.**

| Model | FB15K-237 | | YAGO3-10 | | WN18RR | |
|---|---|---|---|---|---|---|
| | AUC | Specificity | AUC | Specificity | AUC | Specificity |
| CKRL (LT) | .5391 | .0812 | .5035 | .0071 | .5540 | .1236 |
| TransE | .6061 | .2128 | .5772 | .1607 | .5002 | .0005 |
| NoiGAN-TransE (soft) | .6881 | .4788 | .7752 | .5989 | .6036 | .3235 |
| NoiGAN-TransE (hard) | **.7360** | **.6154** | **.8681** | **.7735** | **.6227** | .4114 |
| RotatE | .5373 | .1171 | .5086 | .0176 | .5001 | .0003 |
| NoiGAN-RotatE (soft) | .5848 | .1867 | .6550 | .4372 | .5700 | **.4258** |
| NoiGAN-RotatE (hard) | .5836 | .1868 | .6622 | .4355 | .5860 | .4072 |

**Table 1: Evaluation Results on Noise Detection with Noisy Triples to be 40% of True Triples**

| | Model | FB15K-237 | | | YAGO3-10 | | | WN18RR | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | MRR | H@1 | H@10 | MRR | H@1 | H@10 | MRR | H@1 | H@10 |
| Without Noise | DistMult | .218 | .136 | .388 | .292 | .203 | .470 | .424 | .382 | .507 |
| | KBGAN | .266 | .186 | .427 | .071 | .041 | .124 | .215 | .036 | .507 |
| | Attention | **.436** | **.360** | **.589** | - | - | - | - | .443 | **.583** |
| | TransE | .330 | .229 | .531 | .410 | .299 | **.620** | .229 | .022 | .526 |
| | CKRL (LT) | .330 | .229 | .529 | .377 | .264 | .585 | .231 | .027 | .530 |
| | NoiGAN-TransE (soft) | .318 | .217 | .518 | .370 | .267 | .567 | .231 | .026 | .524 |
| | NoiGAN-TransE (hard) | .308 | .209 | .502 | .345 | .248 | .531 | .223 | .031 | .503 |
| | RotatE | .326 | .228 | .521 | .380 | .269 | .594 | **.472** | **.429** | .557 |
| | NoiGAN-RotatE (soft) | .326 | .229 | .520 | **.430** | **.330** | **.620** | .466 | .420 | .554 |
| | NoiGAN-RotatE (hard) | .325 | .228 | .518 | .413 | .312 | .602 | .465 | .420 | .551 |
| With Noise | DistMult | .196 | .122 | .342 | .074 | .030 | .160 | .328 | .266 | .420 |
| | KBGAN | .123 | .040 | .288 | .026 | .016 | .040 | .161 | .007 | .429 |
| | Attention | .255 | .174 | .414 | - | - | - | .192 | .012 | .453 |
| | TransE | .167 | .019 | .440 | .085 | .005 | .242 | .169 | .005 | .439 |
| | CKRL (LT) | .171 | .021 | .449 | .090 | .004 | .262 | .169 | .007 | .441 |
| | NoiGAN-TransE (soft) | .189 | .037 | .464 | .124 | .003 | .358 | .166 | .002 | .442 |
| | NoiGAN-TransE (hard) | .199 | .059 | .457 | **.210** | **.091** | **.443** | .152 | .004 | .392 |
| | RotatE | .254 | .154 | .444 | .037 | .001 | .103 | .357 | .292 | .456 |
| | NoiGAN-RotatE (soft) | .279 | .179 | .475 | .121 | .034 | .296 | .366 | .295 | .473 |
| | NoiGAN-RotatE (hard) | **.282** | **.184** | **.478** | .128 | .040 | .310 | **.368** | **.298** | **.474** |

**Table 2: Evaluation Results on KG Completion with Noisy Triples to be 70% of True Triples**




(a) Correct Triples. (b) Noise.

**Figure 3: Distribution of Confidence Score on YAGO3-10.**

method claims that they can ensure robust performance, its performance drops rapidly once the noise is introduced, which shows that our NoiGAN significantly outperforms it in terms of robustness.

**Robustness Analysis of the NoiGAN.** To further analyze the robustness of the NoiGAN, we investigate how performance change with noisy triples to be different ratio (e.g., 40%, 70% and 100%) of original data on YAGO3-10 dataset for both noise detection and KG completion tasks. We divided the methods into two categories according to their score functions, including (1) TransE based methods, which consists of TransE, CKRL (LT) and NoiGAN-TransE; and (2) RotatE based methods, which consists of RotatE and NoiGAN-RotatE. The results of KG noise detection and KG completion are presented in Figure 2 (a) and Figure 2 (b) respectively. We can observed that: (1) For noise detection task, hard version of NoiGAN consistently outperforms soft version of NoiGAN in all cases. However, soft version of NoiGAN show better robustness w.r.t the increase of noise. (2) For KG completion task, if we do not introduce noise, the performance of NoiGAN-RotatE is almost the same as its baselines while the performance of NoiGAN-TransE is a little worse than its baseline. As the amount of added noise rises, the improvement introduced by the NoiGAN becomes more significant, which further proves the robustness of the NoiGAN.

**Case Study.** To demonstrate the power of discriminator in distinguishing noise, we conduct experiments over YAGO3-10 with 40% noise on NoiGAN-TransE (soft) and display confidence scores of triples in form of histogram. The distribution for correct triples are shown in Figure 3(a) while the distribution for noise can be found in Figure 3(b). We can see that confidence scores of the correct triples are mainly in the region of $[0.8, 1]$. Instead, confidence scores of the noise are more evenly distributed, which are more concentrated in the region of $[0, 0.2]$. The results indicate that the task of classifying the correct triples is much easier than the task of classifying the noise, which also validates our assumption that deep learning models tend to memorize correct triples at first and thus leads to the great generalization performance of correct triples.

## 5 RELATED WORK

**Knowledge Graph Completion.** Embedding based methods currently hold the state-of-the-art in KG completion for their promising results [24]. They aim to capture the similarity of entities by embedding entities and relations into continuous low-dimensional vectors. Existing methods can be roughly divided into two categories: translational distance models and semantic matching models. Translational distance models measure the plausibility of a fact as the distance between two entities after a translation carried out by the relation. Some representative approaches include TransE [1], TransH [25] and TransR [13]. Semantic matching models measure plausibility of facts by matching latent semantics of entities and relations embodied in their vector space representations. The typical models include RESCAL [16], DistMult [28] and ComplEx [21]. To optimize the KGE model, negative sampling is usually required to minimize the margin based ranking loss. A conventional method to construct negative samples is randomly sampling. However, negative samples generated through a random mode are often too easy to be discriminated from positive facts and thus make little contribute towards the training. Some recent work proposed to incorporate GAN for better negative sampling to improve the quality of embeddings [2, 23]. Nonetheless, none of the above methods has considered the potential involvement of noisy data in KGs, which leads to their sensitivity to unreliable data [18]. In this paper, we proposed a novel technique to enable current embedding models to cope with noisy data.

**Error Detection in Knowledge Graph.** Due to the lack of noisy data samples, error detection in knowledge graph is a challenging task. Existing methods can be broadly divided into two categories, including ontology based methods and anomaly detection based methods. Ontology based methods address this problem by exploring additional ontology information. A larger number of ontology reasoners are developed to utilize logic programming to derive uncovering contradictions among observed facts [3, 4, 14]. Rich ontology information is required for such kind of methods and thus impede its application to real-world knowledge graphs. Another kind of methods is anomaly detection based methods [5, 26]. The main drawback of this kind of methods is that they do not necessarily identify errors, but also natural outliers, which will compromise the objectivity of its results.

**Noise Aware Knowledge Graph Embedding.** More recently, detecting noises while learning knowledge representations simultaneously becomes popular. A novel confidence-aware framework [19, 27] proposed to incorporate triple confidence into KGE model to learn noise aware knowledge graph representations simultaneously. However, it measures the triple confidence merely based on the how well the triple fits the model, which is easily affected by model bias. Recently, another measurement [10] proposed to synthesizes the internal semantic information in the triples and the global inference information of the KG to jointly measure the trustworthiness of triples. In particular, it fusions information from three levels for trustworthiness measurement.

## 6 CONCLUSION

In this paper, we propose a novel generative adversarial framework NoiGAN, to learn noise-aware KG embedding. Under the framework, KGE and error detection tasks are joint trained to mutually enhance each other. On one hand, error detection model is useful to provide clean KGs for KGE. On the other hand, KGE can help prepare both positive and negative data samples to classify noisy triples. Extensive experiments show the superiority of NoiGAN in regard to both KG completion task and error detection task.

# REFERENCES


[1] 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.

[2] Liwei Cai and William Yang Wang. 2017. Kbgan: Adversarial learning for knowledge graph embeddings. *arXiv preprint arXiv:1711.04071* (2017).

[3] Kathrin Dentler, Ronald Cornet, Annette Ten Teije, and Nicolette De Keizer. 2011. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web* 2, 2 (2011), 71–87.

[4] Li Ding, Pranam Kolari, Zhongli Ding, and Sasikanth Avancha. 2007. Using ontologies in the semantic web: A survey. In *Ontologies*. Springer, 79–113.

[5] Daniel Fleischhacker, Heiko Paulheim, Volha Bryl, Johanna Völker, and Christian Bizer. 2014. Detecting errors in numerical linked data using cross-checked outlier detection. In *International semantic web conference*. Springer, 357–372.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[7] Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. 2005. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In *Proceedings of the 31st international conference on very large data bases*. VLDB Endowment, 529–540.

[8] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Advances in neural information processing systems*. 8527–8537.

[9] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 221–231.

[10] Shengbin Jia, Yang Xiang, Xiaojun Chen, and Kun Wang. 2019. Triple Trustworthiness Measurement for Knowledge Graph. In *The World Wide Web Conference*. 2865–2871.

[11] Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*. 4284–4295.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[13] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[14] Marko Luther, Thorsten Liebig, Sebastian Böhm, and Olaf Noppens. 2009. Who the Heck is the Father of Bob?. In *European Semantic Web Conference*. Springer, 66–80.

[15] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. *arXiv preprint arXiv:1906.01195* (2019).

[16] 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.

[17] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.

[18] Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1751–1756.

[19] Yingchun Shan, Chenyang Bu, Xiaojian Liu, Shengwei Ji, and Lei Li. 2018. Confidence-Aware Negative Sampling Method for Noisy Knowledge Graph Embedding. In *2018 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 33–40.

[20] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[21] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.

[22] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 515–524.

[23] Peifeng Wang, Shuangyin Li, and Rong Pan. 2018. Incorporating GAN for negative sampling in knowledge representation learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[24] 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* 29, 12 (2017), 2724–2743.

[25] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[26] Dominik Wienand and Heiko Paulheim. 2014. Detecting incorrect numerical data in dbpedia. In *European Semantic Web Conference*. Springer, 504–518.

[27] Ruobing Xie, Zhiyuan Liu, Fen Lin, and Leyu Lin. 2018. Does William Shakespeare really write Hamlet? knowledge representation learning with confidence. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[28] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[29] Wentau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *IJCNLP*. Beijing, China, 1321–1331.

| Dataset | #Relations | #Entities | #Train | #Valid | #Test |
|---|---|---|---|---|---|
| FB15K-237 | 237 | 14,541 | 272,115 | 17,535 | 20,466 |
| WN18RR | 11 | 40,943 | 86,835 | 3,034 | 3,134 |
| YAGO3-10 | 37 | 123,182 | 1,079,040 | 5,000 | 5,000 |

**Table 3: Data Statistics**

| Model | Dataset | Batch Size | # Negative Samples | Embedding Dim | $\gamma$ | $\alpha$ | Learning Rate |
|---|---|---|---|---|---|---|---|
| NoiGAN-TransE | FB15K-237 | 1024 | 256 | 1000 | 24 | 1 | 0.0001 |
| | WN18RR | 512 | 1024 | 500 | 6 | 0.5 | 0.00005 |
| | YAGO3-10 | 512 | 1024 | 250 | 12 | 0.5 | 0.001 |
| NoiGAN-RotatE | FB15K-237 | 1024 | 256 | 250 | 9 | 1 | 0.00005 |
| | WN18RR | 512 | 1024 | 250 | 6 | 0.5 | 0.00005 |
| | YAGO3-10 | 1024 | 400 | 250 | 24 | 1 | 0.0002 |

**Table 4: The best hyperparameter setting of NoiGAN on several benchmarks.**

# 7 APPENDIX

# 8 STATISTICS OF THE DATASETS

The statistics of all datasets are summarized in Table 3. These benchmark datasets benefit from human curation that results in highly reliable facts. To simulate the real-world knowledge graphs extracted automatically from unstructured text data, we modified these benchmark datasets to include noisy triples. As all kinds of noise might be contained when we construct knowledge graphs, our approach to introducing noise is to substitute the true head entity or tail entity with any randomly selected entity. Following this approach, we construct three KGs based on each benchmark dataset with noisy triples to be different ratio (e.g., 40%, 70% and 100%) of original data. All noisy datasets share the same entities, relations, validation and test sets with the original benchmark dataset, with all generated noisy triples fused into the original training set.

# 9 HYPERPARAMETER SETTINGS

Adam [12] is adopted as the optimizer. We set the parameters for all methods by a grid search strategy. The range of different parameters is set as follows: embedding dimension $k \in \{250, 500, 1000\}$, batch size $b \in \{256, 512, 1024\}$, and fixed margin $\gamma \in \{6, 9, 12, 24\}$. Afterwards, we compare the best results of different methods. Both the entity embeddings and the relation embeddings are uniformly initialized and no regularization is imposed on them. As mention in Section 3.2, we implement both discriminator and the generator as simple two-layer fully connected neural networks. The size of hidden states for each of the two networks is set to be 10. The detailed hyperparameter settings can be found in Table 4.