Data Poisoning Attack against Knowledge Graph Embedding

Hengtong Zhang[1], Tianhang Zheng[1], Jing Gao[1], Chenglin Miao[1], Lu Su[1], Yaliang Li[2], Kui Ren[3]
[1]SUNY at Buffalo, Buffalo, NY USA
[2]Alibaba Group, Bellevue, WA USA
[3]Zhejiang University, Zhejiang, China
{hengtong, tzheng4, jing, cmiao, lusu}@buffalo.edu, yaliang.li@alibaba-inc.com, kuiren@zju.edu.cn

Abstract

Knowledge graph embedding (KGE) is a technique for learning continuous embeddings for entities and relations in the knowledge graph. Due to its benefit to a variety of downstream tasks such as knowledge graph completion, question answering and recommendation, KGE has gained significant attention recently. Despite its effectiveness in a benign environment, KGE's robustness to adversarial attacks is not well-studied. Existing attack methods on graph data cannot be directly applied to attack the embeddings of knowledge graph due to its heterogeneity. To fill this gap, we propose a collection of data poisoning attack strategies, which can effectively manipulate the plausibility of arbitrary targeted facts in a knowledge graph by adding or deleting facts on the graph. The effectiveness and efficiency of the proposed attack strategies are verified by extensive evaluations on two widely-used benchmarks.

1 Introduction

Knowledge graphs have become a critical resource for a large collection of real world applications, such as information extraction [Mintz *et al.*, 2009], question answering [Yih *et al.*, 2015] and recommendation system [Zhang *et al.*, 2016]. Due to its wide application domains, both academia and industry have spent considerable efforts on constructing large-scale knowledge graphs, such as YAGO [Hoffart *et al.*, 2013], Freebase [Bollacker *et al.*, 2008], and Google Knowledge Graph[1]. In knowledge graphs, knowledge facts are usually stored as (*head entity, relation, tail entity*) triples. For instance, the fact triple *(Albert Einstein, Profession, Scientist)* means that Albert Einstein's profession is a scientist.

Although such triples can effectively record abundant knowledge, their underlying symbolic nature makes them difficult to be directly fed to many machine learning models. Hence, knowledge graph embedding (KGE), which projects the symbolic entities and relations into continuous vector space, has quickly gained significant attention [Nickel *et al.*, 2011; Lin *et al.*, 2015; Bordes *et al.*, 2013; Yang *et al.*, 2014b; Trouillon *et al.*, 2016]. These compact embeddings can preserve the inherent characteristics of entities and relations while enabling the use of these knowledge facts for a large variety of downstream tasks such as link prediction, question answering, and recommendation.

Despite the increasing success and popularity of Knowledge graph embeddings, their robustness has not been fully analyzed. In fact, many knowledge graphs are built upon unreliable or even public data sources. For instance, the well known *Freebase* harvests its data from various sources including individual, user-submitted wiki contributions[2]. The openness of such data unfortunately would make KGE vulnerable to malicious attacks. When being attacked, substantial unreliable or even biased knowledge graph embeddings would be generated, leading to serious impairment and financial loss of many downstream applications. For instance, a variety of recommendation algorithms (e.g., [Zhang *et al.*, 2016; Wang *et al.*, 2018]) utilize KGEs of products as external references. If KGEs are manipulated, the recommendation results will be biased. This phenomenon can largely hurt user experiences. Therefore, there is a strong need for the analysis of the vulnerability of knowledge graph embeddings.

In this paper, for the first time, we systemically investigate the vulnerability of KGE, through designing efficient adversarial attack strategies. Due to the unique characteristics of knowledge graph and its embedding models, existing adversarial attack methods on graph data [Zügner *et al.*, 2018; Sun *et al.*, 2018; Bojcheski and Günnemann, 2018] cannot be directly applied to attack KGE methods. First, they are all designed for homogeneous graphs, in which there is only a single type of nodes or links. However, in a knowledge graph, both the entities (nodes) and the relations (links) between entities are of different types. Second, existing attack methods for homogeneous graphs usually have strict requirements on the formulation of the targeted methods. For instance, the attack strategies proposed in [Sun *et al.*, 2018; Bojcheski and Günnemann, 2018] can only work for the embedding methods that can be transformed into matrix factorization. However, the KGE methods are diverse and may not be able to be transformed into matrix factorization problems.

In this paper, we introduce the first study on the vulnerabil-

[1]https://developers.google.com/knowledge-graph/

[2]https://www.nytimes.com/2007/03/09/technology/09data.html

ity of KGE and propose a family of effective *data poisoning attack* strategies against KGE methods. Our proposed attack strategies can guide the adversary to manipulate the training set of KGE by adding and/or deleting some specific facts to promote or degrade the plausibility of specific targeted facts, which can potentially influence a large variety of applications that utilize the knowledge graph. The proposed strategies include both direct scheme which directly manipulates the embeddings of entities involved in the targeted facts and indirect scheme which utilizes other entities as proxies to achieve the attack goal. Empirically, we perform poisoning attack experiments against three most representative KGE methods on two common KGE datasets (FB15K, WN18), and verify the effectiveness of the proposed adversarial attack. Results show that the proposed strategies can dramatically worsen the link prediction results of targeted facts with only a small amount of changes to the graph needed.

2 Related Work

Knowledge Graph Embeddings. KGE as an emerging research topic has attracted tremendous interest. A large number of KGE models have been proposed to represent entities and relations in a knowledge graph with vectors or matrices. RESCAL [Nickel *et al.*, 2011], which is based on bi-linear matrix factorization, is one of the earliest KGE models. Then [Bordes *et al.*, 2013] introduces the first translation-based KGE method TransE. Given a fact (h, r, t), composed of a relation (r) and two entities (h and t) in the knowledge graph, TransE learns vector representations of h, t, and r (i.e., $\mathbf{h}$, $\mathbf{t}$ and $\mathbf{r}$) by compelling $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. Later, a large collection of variants, such as TransH [Wang *et al.*, 2014], TransR [Lin *et al.*, 2015], TransD [Ji *et al.*, 2015] and TransA [Xiao *et al.*, 2015], extend TransE by projecting the embedding vector into various spaces. On the other hand, DistMult [Yang *et al.*, 2014a] simplifies RESCAL by only using a diagonal matrix, and ComPlex [Trouillon *et al.*, 2016] extends DistMult into the complex number field. [Wang *et al.*, 2017] provides a comprehensive survey on these models. The attack strategy proposed in this paper can be used to attack most of the existing KGE models.

Data Poisoning Attack v.s. Evasion Attack. *Data poisoning attacks*, such as those in [Biggio *et al.*, 2012; Steinhardt *et al.*, 2017] are a family of adversarial attacks on machine learning methods. In these works, the attacker can access the training data of the learning algorithm, and has the power to manipulate a fraction of the training data in order to make the trained model meet certain desired objectives. *Evasion attacks* such a those in [Goodfellow *et al.*, 2014; Kurakin *et al.*, 2016] are another prevalent type of attack that may be encountered in adversarial settings. In the evasion setting, malicious samples are generated at test time to evade detection. In this paper, the proposed adversarial attack strategies against KGE methods can be categorized into the *data poisoning attack* setting.

Adversarial Attacks on Graphs. There are limited existing works on adversarial attacks for graph learning tasks: node classification [Zügner *et al.*, 2018; Dai *et al.*, 2018], graph classification [Dai *et al.*, 2018], link prediction [Chen *et al.*, 2018] and node embedding [Sun *et al.*, 2018; Bojchevski and Günnemann, 2018]. The first work, introduced by [Zügner *et al.*, 2018] linearizes the graph convolutional network (GCN) [Kipf and Welling, 2016] to derive the closed-form expression for the change in class probabilities for a given edge/feature perturbation and greedily pick the top perturbations that change the class probabilities. [Dai *et al.*, 2018] proposes a reinforcement learning based approach where the attack agent interacts with the targeted graph/node classifier to learn the policy of selecting the edge perturbations that fool the classifier. [Chen *et al.*, 2018] adopts the fast gradient sign scheme to perform *evasion attack* against the link prediction task with GCN. [Sun *et al.*, 2018] and [Bojchevski and Günnemann, 2018] propose *data poisoning attack* against factorization-based embedding methods on homogeneous graphs. They both formulate the poisoning attack as bilevel optimization problems. The former exploits the eigenvalue perturbation theory [Stewart, 1990], while the latter directly adopts iterative gradient method [Carlini and Wagner, 2017] to solve the problem. To the best of our knowledge, there is no existing investigation on adversarial attack for heterogeneous graphs, in which the links and/or nodes are of different types, like knowledge graphs. This paper sheds first light on this important problem that has not been studied yet.

3 Data Poisoning Attack against Knowledge Graph Embedding (KGE) Methods

Let us consider a knowledge graph $\mathcal{KG}$, with a training set denoted as $\{(e_n^h, r_n, e_n^t)\}_{n=1}^N$ and a targeted fact triple $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ that does not exist in the training set. The goal of the attacker is to manipulate the learned embeddings, which would *degrade* (or *promote*) the plausibility of $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ measured by a specific *fact plausibility scoring* function f. Without loss of generality, we focus on degrading the targeted fact. We also assume that the attacker has a limited attacking budget. *In this paper, the attacking budget is the number of perturbations per target*. Formally, the attack task is defined as follows:

Definition 1 (Problem Definition)**.** *Consider a targeted fact triple $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ that does not exist in the training set, we use $\boldsymbol{e}_x^{h,target}$ to denote the embedding of the head entity $e_x^{h,target}$, $\boldsymbol{e}_x^{t,target}$ to denote the embedding of the tail entity $e_x^{t,target}$ and $\boldsymbol{r}_x^{target}$ to denote the embedding of the relation r_x^{target} from the original training set. Our task is to minimize the plausibility of $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$, i.e., $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})$, by making perturbations (i.e., adding/deleting facts) on the training set. We assume the attacker has a given, fixed budget and is only capable of making M perturbations.*

Due to the discrete and combinatorial nature of the knowledge graph, solving this problem is highly challenging. Intuitively, in order to manipulate the plausibility of a specific targeted fact, we need to shift either the embedding vectors related to its entities or the embedding vectors/matrices related to its relations. However, in a knowledge graph, the number of facts that a relation type involves is much larger than the number of facts that an entity type involves. For instance, in the well-known knowledge graph *Freebase*, the number of entities is over 30 million, while the number of relation types is only 1345. This leads to the fact that the innate *characteristics* of each relation type is far more stable than that of entities and is difficult to be manipulated via a small number of modifications. Hence, in this paper, we focus on manipulating the plausibility of targeted facts from the perspective of entities. To achieve the attack goal, in the rest of this section, we propose a collection of effective yet efficient attack strategies.

3.1 Direct Attack

Given the uncontaminated knowledge graph, the goal of direct attack is to determine a collection of perturbations (i.e., fact adding/deleting actions) to shift the embeddings of the entities involved in the targeted fact to minimize the plausibility of the targeted fact. First, we determine the optimal shifting direction that the entity's embedding should move towards. Then we rank the possible perturbation actions by analyzing the training process of KGE models and designing scoring functions, which estimate the benefit of a perturbation, i.e., how much shifting can be achieved by this perturbation along the desired direction. We name the score as *perturbation benefit score* and calculate such score for every possible perturbation. Finally, we conduct the Top-M perturbations with highest perturbation benefit scores, where M is the attack budget.

Suppose we want to *degrade* the plausibility of the fact $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$. For simplicity, let's focus on shifting the embedding of one of the entities in $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$, say head entity $e_x^{h,target}$, from $\boldsymbol{e}_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$, without loss of generality. Here, $\boldsymbol{\epsilon}_x^*$ denotes the *embedding shifting vector*. The fastest direction of decreasing $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})$ is opposite to its partial derivative with respect to $\boldsymbol{e}_x^{h,target}$. Let ϵ_h be the perturbation step size, the optimal embedding shifting vector is:

$$\boldsymbol{\epsilon}_x^* = -\epsilon_h \cdot \frac{\partial f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_x^{target}, \boldsymbol{e}_x^{t,target})}{\partial \boldsymbol{e}_x^{h,target}}. \tag{1}$$

As mentioned in the problem definition, in order to shift $\boldsymbol{e}_x^{h,target}$ by $\boldsymbol{\epsilon}_x^*$, the adversary is allowed to add perturbation facts to the knowledge graph or delete facts from the knowledge graph. Given the optimal embedding shifting vector $\boldsymbol{\epsilon}_x^*$, we then find a ranking of the all the perturbation (add or delete) candidates. We discuss the two schemes in detail as follows.

Direct Deleting Attack. Consider the uncontaminated training set, under the direct adversarial attack scheme, in order to shift the embedding of $e_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$, we need to select and delete one or more facts that directly involve entity $e_x^{h,target}$. Intuitively, the fact to delete should have a great influence on the embedding of $e_x^{h,target}$, while at the same time not hinder the process of shifting the embedding of $e_x^{h,target}$ to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$. To design a scoring criterion that captures these intuitions, let us look into the training process of KGE model. Consider the specific deletion candidate $(e_x^{h,target}, r_i, e_i^t)$ that involves $e_x^{h,target}$. During training, the sum of the fact plausibility scores of the observed training samples is maximized. On one hand, the more plausible the fact $(e_x^{h,target}, r_i, e_i^t)$ is, the more it contributes to the final embedding of $e_x^{h,target}$. Hence, the perturbation benefit score of deleting $(e_x^{h,target}, r_i, e_i^t)$ should be proportional to $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$. On the other hand, if the plausibility of fact $(e_x^{h,target}, r_i, e_i^t)$ is large after $\boldsymbol{e}_x^{h,target}$ is shifted to $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$ (i.e., $f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$ is large), it means that the fact $(e_x^{h,target}, r_i, e_i^t)$ has a great positive impact on the embedding shifting and should not be deleted. Hence, the perturbation benefit score of deleting $(e_x^{h,target}, r_i, e_i^t)$ should be inversely proportional to $f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t)$. Formally, let the set of all the delete candidates be: $\mathcal{D}_{\mathcal{D}} = \{(e_i^h, r_i, e_i^t) \mid e_i^h = e_x^{h,target} \text{ and } (e_i^h, r_i, e_i^t) \in \mathcal{KG}\}$, which intuitively denote the set of facts that involve $e_x^{h,target}$ as the head entity in the training set. The perturbation benefit score of deleting a specific perturbation fact $(e_x^{h,target}, r_i, e_i^t)$ can be estimated as:

$$\begin{aligned}\eta^-(e_x^{h,target}, r_i, e_i^t) =& f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_i, \boldsymbol{e}_i^t) \\ &- \lambda_1 f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_i, \boldsymbol{e}_i^t),\end{aligned} \tag{2}$$

where $\boldsymbol{e}_x^{h,target}$, $\boldsymbol{r}_i$, and $\boldsymbol{e}_i^t$ denote the embeddings of $e_x^{h,target}$, r_i and e_i^t on the uncontaminated training set.

Direct Adding Attack. Now we discuss how to conduct direct adding perturbation. To shift the embedding of $e_{x^{h,target}}$ by $\boldsymbol{\epsilon}^*$, we just need to add new facts that involve $e_x^{h,target}$ to make $f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_j, \boldsymbol{e}_j^t)$ less plausible. The set of all the possible adding candidates can be denoted as $\mathcal{D}_{\mathcal{A}} = \{e_x^{h,target}\} \times \{(r_j, e_j^t) \mid \forall r_j \in \mathcal{KG} \text{ and } e_j^t \in \mathcal{KG}\}$, where $\{(r_j, e_j^t) \mid \forall r_j \in \mathcal{KG} \text{ and } e_j^t \in \mathcal{KG}\}$ denotes all the possible "relation-tail entity" combinations in the knowledge graph and $\times$ stands for Cartesian product. In practice, for better efficiency, we can downsample a subset from all the possible "relation-tail entity" combinations. Formally, the perturbation benefit score of a specific candidate to add (i.e., $(e_x^{h,target}, r_j, e_j^t)$) can be estimated as:

$$\begin{aligned}\eta^+(e_x^{h,target}, r_j, e_j^t) =& \lambda_2 f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_j, \boldsymbol{e}_j^t) \\ &- \lambda_3 f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_j, \boldsymbol{e}_j^t),\end{aligned} \tag{3}$$

where $\boldsymbol{e}_x^{h,target}$, $\boldsymbol{r}_j$, and $\boldsymbol{e}_i^t$ denote the embeddings on the uncontaminated training set.

3.2 Indirect Attack

Although the direct attack strategy is intuitive and effective, it is possible to be detected by data sanity check. In this section, we move on to introduce a more complicated yet more stealthy adversarial attack scheme, i.e., indirect attack. Suppose a KGE user want to query the plausibility of a potential fact (h, r, t). Due to the huge scales of real-world knowledge graphs, even in the most optimistic situation, we may merely carry data sanity test on the facts related to h and t. However, for indirect attack, instead of adding or deleting the facts that involve the entities in the targeted fact, we propose to perturb the facts that involve other entities in the knowledge graph and let the perturbation effect propagate to the targeted fact. Thus, detecting these perturbations requires data sanity tests on facts that involves every entity that are hops away from h and t. When the number of hops increases linearly, the data sanity cost will have a exponential growth. Even though there is an Oracle that can find these anomalous facts effectively, defenders cannot determine the targeted fact(s) of these perturbations. For a better description, we provide the following toy example, which is used throughout this section.

Example 1. *Suppose we want to degrade the plausibility of the targeted fact* $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$ *via shifting the embedding of the targeted entity* $e_x^{h,target}$ *by* $\boldsymbol{\epsilon}_x^*$*, without loss of generality. Under indirect attack scheme, we perturb the facts that involve the K-hop neighbors of* $e_x^{h,target}$*. These K-hop neighbors are called* proxy entities*. Then the entities between the K-hop neighbors (proxy entities) and* $e_x^{h,target}$ *are* intermediate entities *to propagate the influence of the perturbations to* $e_x^{h,target}$*. The propagation path can be illustrated as follows:*

$$e_x^{h,target} \xleftrightarrow{r_{x,1}} e_{x,1} \xleftrightarrow{r_{x,2}} e_{x,2} \cdots \xleftrightarrow{r_{x,K}} e_{x,K}$$

where we use $\xleftrightarrow{r_{x,\cdot}}$ *to denote the directional relation and use notation* $e_{x,\cdot}$ *to denote the entities on the path. A specific* $e_{x,\cdot}$ *can work as both the head entity and the tail entity. The notations in the path above are adopted in the rest of this section.*

When the perturbations on the proxy entity cause an embedding shift on itself, the embeddings of its neighboring entities will also be influenced. The influence will propagate back to the embedding of the targeted entity ultimately.

However, finding the effective perturbations on the proxy entities, which are K-hop away from the targeted entity, is indeed a challenging task. The task involves two key problems: **(1)** *Given a specific propagation path, how can we determine the desired embedding shifting vectors on its intermediate entities and its proxy entity, in order to accomplish the embedding shifting goal on the targeted entity?* **(2)** *How do we select the propagation paths to propagate the influence of perturbation to the targeted entity?* In the rest of this section, we discuss strategies to solve these key problems and propose a criterion to evaluate the benefit of an indirect perturbation (i.e., the *perturbation benefit score*).

For the first problem, given a specific path, in order to conduct a perturbation that makes the embedding of $e_x^{h,target}$ shift towards the desired direction (i.e., the direction of $\boldsymbol{\epsilon}_x^*$), we decide the shifting goal for each entity on the path in a *recurrent* way. Suppose we want to shift $\boldsymbol{e}_x^{h,target}$ by $\boldsymbol{\epsilon}_x^*$ via the intermediate entities along the path specified in Example 1. The entity that directly influences $\boldsymbol{e}_x^{h,target}$ is its neighbor $e_{x,1}$ and what we need to do is to determine the ideal embedding shifting vector $\boldsymbol{\epsilon}_{x,1}^*$ on $e_{x,1}$, so that the desired embedding shift on $e_x^{h,target}$ (i.e., $\boldsymbol{\epsilon}_x^*$) is approached to the greatest extent. Formally, $\boldsymbol{\epsilon}_{x,1}^*$ should satisfy:

$$\begin{aligned}\boldsymbol{\epsilon}_{x,1}^* = \arg\max_{\boldsymbol{\epsilon}} & f(\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*, \boldsymbol{r}_{x,1}, \boldsymbol{e}_{x,1} + \boldsymbol{\epsilon}) \\ & - f(\boldsymbol{e}_x^{h,target}, \boldsymbol{r}_{x,1}, \boldsymbol{e}_{x,1} + \boldsymbol{\epsilon}) \\ & s.t. \ ||\boldsymbol{\epsilon}||_2 = \epsilon_h,\end{aligned} \tag{4}$$

where ϵ_h is the perturbation step size, $\boldsymbol{e}_{x,1}$ denotes the embedding of $e_{x,1}$, and $\boldsymbol{r}_{x,1}$ denotes the embedding of $r_{x,1}$. As

Algorithm 1 Indirect Attack

```
Require: Targeted fact $(e_x^{h,target}, r_x^{target}, e_x^{t,target})$, Neighbor hop $K$, Targeted entity $e_x^{h,target}$.
 1: Exhaust all the $K$-hop paths originating from $e_x^{h,target}$
 2: Exhaust all the possible perturbation candidates on the proxy entities of these $K$-hop paths.
 3: Calculate $\boldsymbol{\epsilon}_x^*$ for $e_x^{h,target}$ according to Eq. (1).
 4: for each path $k$ do
 5:     for each intermediate entity / proxy entity $e_{x,i}$ in path $x$ do
 6:         Calculate $\boldsymbol{\epsilon}_{x,i}^*$ according to Eq. (4).
 7:     end for
 8:     Calculate the score $\psi$ for each perturbation on the current proxy entity according to Eq. (2), Eq. (3) and (5).
 9: end for
10: Select $M$ perturbations with highest scores (i.e., $\psi$) and conduct the attack.
```

a result, the embedding of $e_x^{h,target}$ will have a larger tendency to move towards $\boldsymbol{e}_x^{h,target} + \boldsymbol{\epsilon}_x^*$ than towards $\boldsymbol{e}_x^{h,target}$, during the training process on the contaminated training data. When $\boldsymbol{\epsilon}_{x,1}^*$ is determined, we can further get the embedding shifting vector for $e_{x,2}, \cdots, e_{x,K}$, which are denoted as $\boldsymbol{\epsilon}_{x,2}^*, \cdots, \boldsymbol{\epsilon}_{x,K}^*$, respectively. This process is similar as above.

With the embedding shifting vectors on the proxy entities of each path determined, we calculate the scores η^- and η^+, defined in Eq. (2) and (3) for all the possible add/delete perturbations. These scores are later used to calculate the perturbation benefit score under indirect attack schemes.

For the second problem, we look into the training objective function. Suppose we want to shift the embedding of $e_{x,k-1}$ via its neighbor $e_{x,k}$, when the embedding shift on $\boldsymbol{e}_{x,k}$ is $\boldsymbol{\epsilon}_{x,k}^*$. To estimate the influence of such embedding shift on $\boldsymbol{e}_{x,k-1}$, we isolate all the facts that involve $e_{x,k-1}$ in the training objective function, force a embedding shift $\boldsymbol{\epsilon}_{x,k}$ on $\boldsymbol{e}_{x,k}$ and ignore the negative sampling terms. Formally, the objective function becomes: $\min_{\boldsymbol{e}_{e_{x,k-1}}} \sum_{(e_i^h, r_i, e_i^t) \in D_{e_{x,k-1}}^{\setminus e_{x,k}}} \mathcal{L}(\boldsymbol{e}_i^h, \boldsymbol{r}_i, \boldsymbol{e}_i^t) + \mathcal{L}(\boldsymbol{e}_{x,k-1}, \boldsymbol{r}_{x,k}, \boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k})$, where $D_{e_{x,k-1}}^{\setminus e_{x,k}}$ stands for the set of all the observed facts, which involve $e_{x,k-1}$ except the fact $(e_{x,k-1}, r_{x,k}, e_{x,k})$, in the training set. $\mathcal{L}$ denotes the loss function for a single fact. $\boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k}$ in $\mathcal{L}(\boldsymbol{e}_{x,k-1}, \boldsymbol{r}_{x,k}, \boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k})$ indicates that the embedding of $e_{x,k}$ is already shifted. Clearly, if we fix the embeddings of all the relations and entities except $e_{x,k-1}$, the impact of shifting $\boldsymbol{e}_{x,k}$ to $\boldsymbol{e}_{x,k} + \boldsymbol{\epsilon}_{x,k}$ is highly correlated with the number of facts that involves $e_{x,k-1}$, i.e., $|D_{e_{x,k-1}}|$. That is to say, the more neighbors an entity has, the *less* it will be influenced by a specific perturbation on one of its neighbors.

Based on above discussions, we propose an empirical scoring function to evaluate the perturbation benefit score of every possible perturbation. We still consider the scenario specified in Example 1. Suppose we conduct an add/delete perturbation $(e_{x,K}, r_{x,K}, e_{x,K+1})$ on the proxy entity $e_{x,K}$. The perturbation benefit score of this indirect perturbation is defined as:

$$\begin{aligned}&\psi(e_{x,K}, r_{x,K}, e_{x,K+1})\\ &= \eta(e_{x,K}, r_{x,K}, e_{x,K+1}) - \lambda \log\big(\frac{1}{K}\sum_{k=1}^{K-1}|D_{e_{x,k-1}}| \\ &\quad + \max(\{|D_{e_{x,k-1}}|\}_{k=1}^{K-1})\big),\end{aligned} \quad (5)$$

where $\max(\{|D_{e_{x,k-1}}|\}_{k=1}^{K-1})$ stands for the maximum number of facts that involves each entity k on the path. η is the same as η^+ under add perturbation scheme and is the same as η^- under delete perturbation scheme. λ is a trade-off parameter. The first term estimates the direct perturbation benefit of the perturbation in terms of shifting the proxy entity as desired. The second term evaluates the capability of the intermediate entities on the path in terms of propagating the influence to the targeted entity. As the influence may be diluted by the facts that involve each entity $e_{x,k}$ on the path. A smaller averaged number of facts that involves each entity $e_{x,k}$ on the path indicates a larger capability of the path in terms of propagating the influence. Moreover, we also consider the maximum number of facts that involves each entity $e_{x,k}$ on the path. This is to avoid the case when some intermediate entities, whose embedding is difficult to shift, "block" the propagation path. *In practices, we can first utilize the second term to determine the best P paths in terms of propagating the influence from proxy entities to the targeted entity and then choose what facts to add or delete upon these proxy entities in the best P paths.* [3] The overall workflow of indirect attack is illustrated in Algorithm 1.

4 Experiments

4.1 Datasets and Settings

Datasets. In this paper, we use two common KGE benchmark datasets for our experiment: FB15k and WN18. FB15k is a subset of Freebase, which is a large collaborative knowledge base consisting of a large number of real-world facts. WN18 is a subset of Wordnet [4], which is a large lexical knowledge graph. Both FB15k and WN18 are first introduced by [Bordes *et al.*, 2013]. The training set and the test set of these two datasets are already fixed. *We randomly sample 100 samples in the test set as the targeted facts for the proposed attack strategies.*

Baseline & Targeted Models. Since there are no existing methods that can work under the setting of this paper, we compare the proposed attack schemes with several naive baseline strategies. Specifically, we design *random-dd* (random direct deleting), *random-da* (random direct adding), *random-id* (random indirect deleting), *random-ia* (random indirect adding) as comparison baselines for our proposed *direct deleting attack*, *direct adding attack*, *indirect deleting attack*, *indirect adding attack*, respectively. The difference between the baseline and its corresponding proposed methods is that the perturbation facts to add/delete are randomly selected. For the targeted KGE models, we choose three most representative *TransE* [Bordes *et al.*, 2013], *TransR* [Lin *et al.*, 2015] and *RESCAL* [Nickel *et al.*, 2011] as attack targets.

[3]This strategy is used in the experiments of this paper.

[4]https://wordnet.princeton.edu/

		Clean		random-da		Direct Add	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.24	0.46	0.24	0.42
	TransR	0.24	0.52	0.23	0.42	0.21	0.41
	RESCAL	0.19	0.42	0.20	0.40	0.17	0.39
WN18	TransE	0.39	0.70	0.30	0.68	0.21	0.53
	TransR	0.44	0.73	0.41	0.71	0.22	0.51
	RESCAL	0.41	0.72	0.44	0.69	0.30	0.57

Table 1: Overall Results of Direct Adding Attack

		Clean		random-dd		Direct Delete	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.26	0.54	0.19	0.37
	TransR	0.24	0.52	0.25	0.49	0.18	0.41
	RESCAL	0.19	0.42	0.19	0.38	0.13	0.30
WN18	TransE	0.39	0.70	0.36	0.71	0.11	0.26
	TransR	0.44	0.73	0.43	0.68	0.11	0.24
	RESCAL	0.41	0.72	0.40	0.67	0.02	0.05

Table 2: Overall Results of Direct Deleting Attack

Metrics. In order to evaluate the effectiveness of the proposed attack strategies. We compare the plausibility change of the targeted fact before and after the adversarial attack. Specifically, we follow the evaluation protocol of KGE models described in the previous works like [Bordes *et al.*, 2013]. Given a targeted fact (e_h, r, e_t), we remove the head or tail entity and then replace it with all the possible entities. We first compute plausibility scores of those corrupted facts and then rank them by descending order; the rank of the correct entity is stored. After that, we use *MRR* (Mean Reciprocal Rank of all the ground truth triples) and *H@10* (the proportion of correct entities ranked in top 10, for all the ground truth entities.) as our evaluation metrics. *The smaller MRR and H@10 are on the contaminated dataset, the better the attack performance is.*

Experiment Settings. For the targeted KGE models, we use the standard implementation provided by THUNLP-OpenKE [5] [Han *et al.*, 2018]. The embedding dimension d is fixed to 50. Other parameters of baseline methods are set according to their authors' suggestions. For the proposed attack strategies, the parameter K for indirect attack is fixed to 1. During the experiment all the perturbations are injected into the dataset at the same time. The attack models in this paper are all implemented via Numpy and Python 3.7. The attack models are run on a laptop with 4 GB RAM, 2.7 GHz Intel Core i5 CPU.

4.2 Results and Analysis

In this section, we report and analyze the attack results of the proposed attack strategies under different settings. To avoid confusion, the performance of direct adding attack, direct deleting attack, indirect adding attack, and indirect deleting attack are reported separately in Table 1, 2, 3 and 4.

Overall Attack Performance. Let us first discuss the performances of the direct attack schemes on two datasets. For

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

		Clean		random-ia		Indirect Add	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.25	0.50	0.23	0.47
	TransR	0.24	0.52	0.25	0.51	0.22	0.49
	RESCAL	0.19	0.42	0.19	0.40	0.17	0.36
WN18	TransE	0.39	0.70	0.42	0.71	0.32	0.67
	TransR	0.44	0.73	0.40	0.73	0.34	0.69
	RESCAL	0.41	0.72	0.41	0.69	0.39	0.63

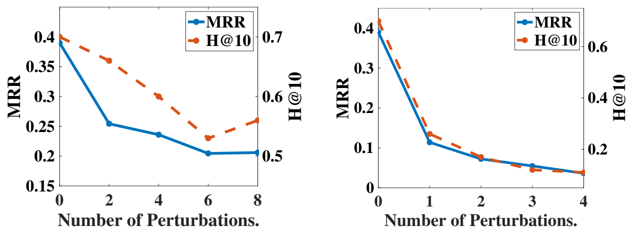
Table 3: Overall Results of Indirect Adding Attack

		Clean		random-id		Indirect Delete	
		MRR	H@10	MRR	H@10	MRR	H@10
FB15K	TransE	0.26	0.49	0.27	0.50	0.22	0.44
	TransR	0.24	0.52	0.25	0.53	0.21	0.48
	RESCAL	0.19	0.42	0.20	0.36	0.16	0.34
WN18	TransE	0.39	0.70	0.44	0.74	0.35	0.68
	TransR	0.44	0.73	0.45	0.74	0.41	0.71
	RESCAL	0.41	0.72	0.42	0.70	0.38	0.64

Table 4: Overall Results of Indirect Deleting Attack

the direct deleting attack scheme, we set the attack budget for each targeted fact to 4 and 1 on FB15K and WN18 dataset, respectively. For the direct attacking attack scheme, the attack budgets for each targeted fact are 8 and 6 for FB15K and WN18 dataset, respectively. These budgets are low enough to make the whole attack process unnoticeable. From the results, we can clearly see that the plausibilities of these targeted facts significantly degrade as desired. We can conclude that these KGE models are quite vulnerable to even a small number of perturbations generated by well-designed attack strategies. For comparison, we have also tested the baseline methods *random-da* and *random-dd*, which cannot achieve satisfactory attack performances. This demonstrates the effectiveness of the proposed strategies. Moreover, we observe that the effectiveness of the proposed strategies is more significant on WN18 dataset than on FB15K dataset. This is because the average number of facts that each entity involves in WN18 dataset is significantly smaller than that in FB15K dataset. Hence, the graph structure of FB15K is more stable and robust. Then, let us move on to the discussion of indirect attack schemes. For the indirect adding attack, we set the attack budget for each targeted fact to 60 and 20 for FB15K and WN18 dataset, respectively. For the indirect deleting attack, the attack budgets for each targeted fact are set to 20 and 5 for FB15K and WN18 dataset, respectively. The reason why indirect attacks need more attack budgets to get comparable results is that only a small portion of the influence caused by the perturbations on proxy entities is propagated to the targeted entity. In contrast, nearly all of the influence of the perturbation is exerted on the targeted entity under direct attack schemes. Like direct attack schemes, these indirect attack schemes also demonstrate their effectiveness. For instance, under the indirect deleting attack scheme, the H@10 and MRR metrics of the targeted facts decrease by approximate 0.03 on FB15K dataset. Thus, the indirect deleting attack schemes can also be used in practices to make the attack process more stealthy.

Analysis of the Number of Perturbations. When conducting the data poisoning attack, one of the most important factors is the number of perturbations (i.e. attack budget). Due to space limit, we merely plot performances of direct attack schemes against TransE w.r.t. the number of perturbations (i.e., attack budgets) on WN18 dataset in Figure 1. From Figure. 1, we can clearly see that the proposed attack strategies consistently degrade the plausibility of the targeted facts under both setting. When the number of perturbations keeps increase, the growth of attack performance becomes slower. This is because when the number of perturbations is small, the selected perturbations are usually of high value in terms of manipulating the plausibility of the targeted facts. When the number of perturbations keeps increase, the high-value perturbations are used up. Hence, the performances become stable.



(a) Direct Adding Attack on WN18 Dataset against TransE (b) Direct Deleting Attack on WN18 Dataset against TransE

Figure 1: Analysis of the Number of Perturbations

Efficiency Analysis. Finally, let us discuss the efficiency of the proposed attack strategies. Here we report the time consumption for the proposed attack strategies to generate the perturbations for a single targeted fact on average. The time consumption of Direct Adding, Direct Deleting, Indirect Adding and Indirect Deleting scheme are 3.36s, 0.13s, 14.04s, and 1.22s, respectively. [6] As one can see, the proposed model takes less than 15 seconds on average to generate the perturbations for a single targeted fact. For the direct deleting attack scheme, the time cost is less than 1 second on average. These results show that the proposed attack strategies are quite efficient.

5 Conclusions

We present the first study on the vulnerability of existing KGE methods and propose a collection of data poisoning attack strategies for different attack scenarios. These attack strategies can be efficiently computed. Experiment results on two benchmark dataset demonstrate that the proposed strategies can effectively manipulate the plausibility of arbitrary facts in the knowledge graph with limited perturbations.

Acknowledgments

We thank our anonymous reviewers for their insightful comments and suggestions on this paper. This work was supported in part by the US National Science Foundation under grants CNS-1742847, IIS-1747614 and CNS-1652503.

[6]Note: Candidate downsampling, which is described in Section 3.1 is used.

References

[Biggio *et al.*, 2012] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

[Bojcheski and Günnemann, 2018] Aleksandar Bojcheski and Stephan Günnemann. Adversarial attacks on node embeddings. *arXiv preprint arXiv:1809.01093*, 2018.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of SIGMOD*, 2008.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in NIPS*, 2013.

[Carlini and Wagner, 2017] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *Proc. of IEEE S&P*, 2017.

[Chen *et al.*, 2018] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

[Dai *et al.*, 2018] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[Han *et al.*, 2018] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proc. of EMNLP Demo*, 2018.

[Hoffart *et al.*, 2013] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[Ji *et al.*, 2015] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proc. of ACL-IJCNLP*, 2015.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Kurakin *et al.*, 2016] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proc. of AAAI*, 2015.

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*, 2009.

[Nickel *et al.*, 2011] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proc. of ICML*, 2011.

[Steinhardt *et al.*, 2017] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. Certified defenses for data poisoning attacks. In *Advances in NIPS*, 2017.

[Stewart, 1990] Gilbert W Stewart. Matrix perturbation theory. 1990.

[Sun *et al.*, 2018] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. Data poisoning attack against unsupervised node embedding methods. *arXiv preprint arXiv:1810.12881*, 2018.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proc. of ICML*, 2016.

[Wang *et al.*, 2014] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proc. of AAAI*, 2014.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE TKDE*, (12):2724–2743, 2017.

[Wang *et al.*, 2018] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proc. of WWW*, pages 1835–1844, 2018.

[Xiao *et al.*, 2015] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transa: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*, 2015.

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

[Yang *et al.*, 2014b] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Learning multi-relational semantics using neural-embedding models. *arXiv preprint arXiv:1411.4072*, 2014.

[Yih *et al.*, 2015] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proc. of ACL-IJCNLP*, 2015.

[Zhang *et al.*, 2016] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proc. of SIGKDD*, 2016.

[Zügner *et al.*, 2018] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on classification models for graphs. *arXiv preprint arXiv:1805.07984*, 2018.