Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods

Peru Bhardwaj[1] **John Kelleher**[2*] **Luca Costabello**[3*] **Declan O'Sullivan**[1*]
[1] ADAPT Centre, Trinity College Dublin, Ireland
[2] ADAPT Centre, TU Dublin, Ireland
[3] Accenture Labs, Ireland
peru.bhardwaj@adaptcentre.ie

Abstract

Despite the widespread use of Knowledge Graph Embeddings (KGE), little is known about the security vulnerabilities that might disrupt their intended behaviour. We study data poisoning attacks against KGE models for link prediction. These attacks craft adversarial additions or deletions at training time to cause model failure at test time. To select adversarial deletions, we propose to use the model-agnostic *instance attribution methods* from Interpretable Machine Learning, which identify the training instances that are most influential to a neural model's predictions on test instances. We use these influential triples as adversarial deletions. We further propose a heuristic method to replace one of the two entities in each influential triple to generate adversarial additions. Our experiments show that the proposed strategies outperform the state-of-art data poisoning attacks on KGE models and improve the MRR degradation due to the attacks by up to 62% over the baselines.

1 Introduction

Knowledge Graph Embeddings (KGE) are the state-of-art models for relational learning on large scale Knowledge Graphs (KG). They drive enterprise products ranging from search engines to social networks to e-commerce (Noy et al., 2019). However, the analysis of their security vulnerabilities has received little attention. Identifying these vulnerabilities is especially important for high-stake domains like healthcare and finance that employ KGE models to make critical decisions (Hogan et al., 2020; Bendtsen and Petrovski, 2019). We study the security vulnerabilities of KGE models through data poisoning attacks (Biggio and Roli, 2018; Joseph et al., 2019) that aim to degrade the predictive performance of learned KGE models by adding or removing triples to the input training graph.

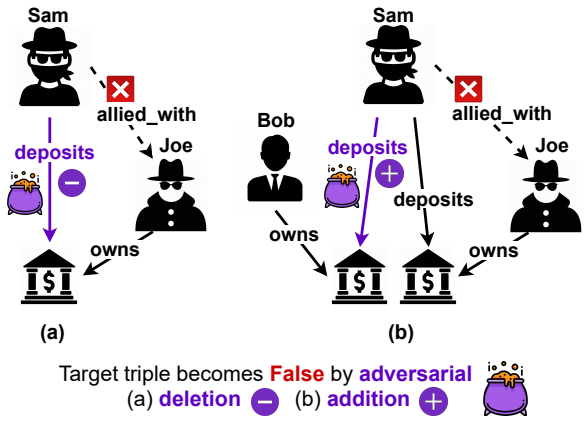

Figure 1: Adversarial attacks against KGE models for fraud detection. The knowledge graph consists of two types of entities - Person and BankAccount. The missing target triple to predict is $(Sam, \texttt{allied\_with}, Joe)$. Original KGE model predicts this triple as True. But a malicious attacker uses the instance attribution methods to either (a) delete an adversarial triple or (b) add an adversarial triple. Now, the KGE model predicts the missing target triple as False.

Designing data poisoning attacks against KGE models poses two main challenges. First, to select adversarial deletions or additions, we need to measure the impact of a candidate perturbation on the model's predictions. But the naive approach of re-training a new KGE model for each candidate perturbation is computationally prohibitive. Second, while the search space for adversarial *deletions* is limited to existing triples in the KG, it is computationally intractable to enumerate through all candidate adversarial *additions*. Furthermore, attack strategies proposed against models for other graph modalities (Xu et al., 2020) do not scale to KGE models; as they would require gradients with respect to a dense adjacency tensor of the KG.

In this work, we propose to use the model-agnostic *instance attribution methods* from Interpretable Machine Learning (Molnar, 2019) to select adversarial deletions and additions against KGE models. Instance attribution methods identify the training instances that are *influential* to

*Equal contribution by last authors.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 8225–8239
November 7–11, 2021.

model predictions, that is, deleting the instances from the training data would considerably change the model parameters or predictions. These methods are widely used to generate post-hoc example-based explanations for deep neural networks on images (Koh and Liang, 2017; Hanawa et al., 2021; Charpiat et al., 2019) and text (Han et al., 2020; Han and Tsvetkov, 2020; Pezeshkpour et al., 2021). Since the KGE models have relatively shallow neural architectures and the instance attribution metrics are independent of the black-box models and the input domain, they are a promising approach to estimate the influence of training triples on the KGE model predictions. Yet, despite their promise, they have not been used on KGE models so far. We use the instance attribution methods to address the challenge of measuring the impact of a candidate adversarial deletion on the model predictions.

We focus on the adversarial goal of *degrading* the KGE model prediction on a given *target triple*. To achieve this goal, we use three types of instance attribution methods - Instance Similarity that compares the feature representations of target and training triples (Hanawa et al., 2021; Charpiat et al., 2019); Gradient Similarity that compares the gradients of model's loss function due to target and training triples (Hanawa et al., 2021; Charpiat et al., 2019); and Influence Function (Koh and Liang, 2017) which is a principled approach from the robust statistics to estimate the effect of removing a training triple on the KGE model's predictions.

Using these metrics, we select the most influential training triple for adversarial deletion. Using the influential triple, we further select adversarial addition by replacing one of the two entities of the influential triple with the most dissimilar entity in the embedding space. The intuition behind this step is to add a triple that would reduce the influence of the influential triple. This solution also overcomes the scalability challenge for adversarial additions by comparing only the entity embeddings to select the replacement. Figure 1 shows an example of the proposed adversarial deletions and additions against KGE models for fraud detection.

We evaluate the proposed attacks for four KGE models - DistMult, ComplEx, ConvE and TransE on two benchmark datasets - WN18RR and FB15k-237. Our results show that instance attribution metrics achieve significantly better performance than *all* state-of-art attacks for both adversarial additions and deletions on three out of four models; and better or equivalent performance on one model. We find that even simple metrics based on instance similarity outperform the state-of-the-art poisoning attacks and are as effective as the computationally expensive Influence Function.

Thus, the main contribution of our research is a collection of effective adversarial deletion and addition strategies based on instance attribution methods against KGE models.

2 Knowledge Graph Embeddings

A Knowledge Graph (KG), is a set of triples $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where each triple encodes the relationship $\mathtt{r}$ as a typed link between the subject entity s and the object entity o, i.e. $\mathcal{T} \coloneqq \{t \coloneqq (s, \mathtt{r}, o) \,|\, s, o \in \mathcal{E}\, and\, \mathtt{r} \in \mathcal{R}\}$. Here, $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations in the knowledge graph. Large scale KGs are often curated automatically from the user content or from the Web and thus are incomplete in practice. To predict the missing links in a KG, the state-of-art method is to learn low dimensional feature vectors for entities and relations in the graph and use them to score the links. These feature vectors are called Knowledge Graph Embeddings (KGE) and denoted as $\boldsymbol{\theta} \coloneqq \{\boldsymbol{E}, \boldsymbol{R}\}$ where $\boldsymbol{E} \in \mathbb{R}^k$ is the embedding matrix for entities, $\boldsymbol{R} \in \mathbb{R}^k$ is the embedding matrix for relations and k is the embedding dimension.

Scoring Functions: KGE models differ from each other by their scoring functions $f : \mathcal{T} \rightarrow \mathbb{R}$ which combine the subject, relation and object embeddings to assign a score to the triple, i.e. $f_t \coloneqq f(\boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \boldsymbol{e}_o)$ where $\boldsymbol{e}_s, \boldsymbol{e}_o \in \boldsymbol{E}$ and $\boldsymbol{e}_\mathtt{r} \in \boldsymbol{R}$. Table 1 shows the different scoring functions of KGE models used in this research.

These scoring functions are used to categorize the models as additive or multiplicative (Chandrahas et al., 2018). *Additive* models apply relation-specific translation from the subject embedding to the object embedding. The scoring function for such models is expressed as $f_t = -\left\|\boldsymbol{M}^1_\mathtt{r}(\boldsymbol{e}_s) + \boldsymbol{e}_\mathtt{r} - \boldsymbol{M}^2_\mathtt{r}(\boldsymbol{e}_o)\right\|$ where $\boldsymbol{M}_\mathtt{r} \in \mathbb{R}^{k \times k}$ is a projection matrix from entity space to relation space. An example of additive models is TransE where $\boldsymbol{M}^1_\mathtt{r} = \boldsymbol{M}^2_\mathtt{r} = \boldsymbol{I}$.

On the other hand, *multiplicative* models score triples through multiplicative interactions between the subject, relation and object embeddings. The scoring function for these models is expressed as $f_t = \boldsymbol{e}_\mathtt{r}^\top \mathcal{F}(\boldsymbol{e}_s, \boldsymbol{e}_o)$ where the function $\mathcal{F}$ measures the compatibility between the subject and

Model	Scoring Function	Feature Vectors
DistMult	$\langle \boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \boldsymbol{e}_o \rangle$	$\boldsymbol{e}_s \circ \boldsymbol{e}_\mathtt{r} \circ \boldsymbol{e}_o$
ComplEx	$\Re(\langle \boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \overline{\boldsymbol{e}_o} \rangle)$	$\Re(\boldsymbol{e}_s \circ \boldsymbol{e}_\mathtt{r} \circ \overline{\boldsymbol{e}_o})$
ConvE	$\langle (\boldsymbol{e}_s * \boldsymbol{e}_\mathtt{r}), \boldsymbol{e}_o \rangle$	$(\boldsymbol{e}_s * \boldsymbol{e}_\mathtt{r}) \circ \boldsymbol{e}_o$
TransE	$-\left\|\boldsymbol{e}_s + \boldsymbol{e}_\mathtt{r} - \boldsymbol{e}_o\right\|_p$	$-(\boldsymbol{e}_s + \boldsymbol{e}_\mathtt{r} - \boldsymbol{e}_o)$

Table 1: Scoring functions f_{sro} and the proposed Triple Feature Vectors $\boldsymbol{f}_{sro}$ of the KGE models used in this research. For ComplEx, $\boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \boldsymbol{e}_o \in \mathbb{C}^k$; for the remaining models $\boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \boldsymbol{e}_o \in \mathbb{R}^k$. Here, $\langle \cdot \rangle$ denotes the tri-linear dot product; $\circ$ denotes the element-wise Hadamard product; $\overline{\cdot}$ denotes conjugate for complex vectors; $\|\cdot\|_p$ denotes l-p norm; $*$ is the neural architecture in ConvE, i.e. $\boldsymbol{e}_s * \boldsymbol{e}_\mathtt{r} := \sigma(\mathrm{vec}(\sigma([\overline{\boldsymbol{e}_\mathtt{r}}, \overline{\boldsymbol{e}_s}] * \boldsymbol{\Omega}))\boldsymbol{W})$ where σ denotes sigmoid activation, $*$ denotes 2D convolution; $\overline{\cdot}$ denotes 2D reshaping of real vectors.

object embeddings and varies across different models within this family. DistMult, ComplEx and ConvE are examples of multiplicative models.

Training: Since the KGs only contain positive triples; to train the KGE model, synthetic negative samples $t' \in \mathcal{T}'$ are generated by replacing the subject or object in the positive triples with other entities in $\mathcal{E}$. That is, for each positive triple $t := (s, \mathtt{r}, o)$, the set of negative samples is $t' := \{(s', \mathtt{r}, o) \cup (s, \mathtt{r}, o')\}$. The training objective is to learn the embeddings that score positive triples existing in the KG higher than the negative triples generated synthetically. To achieve this, a triple-wise loss function $\mathcal{L}(t, \boldsymbol{\theta}) := \ell(t, \boldsymbol{\theta}) + \sum_{t' \in \mathcal{T}'} \ell(t', \boldsymbol{\theta})$ is minimized. Thus, the optimal parameters $\widehat{\boldsymbol{\theta}}$ learned by the model are defined by $\widehat{\boldsymbol{\theta}} := \arg\min_{\boldsymbol{\theta}} \sum_{t \in \mathcal{T}} \mathcal{L}(t, \boldsymbol{\theta})$. Further details on KGE loss functions and negative sampling strategies are available in Ruffinelli et al. (2020).

Missing Link Prediction: Given the learned embeddings $\boldsymbol{\theta}$, missing triples in the knowledge graph are predicted by an entity ranking evaluation protocol. Similar to the training process, subject-side negatives $t'_s = (s', \mathtt{r}, o)$ and object-side negatives $t'_o = (s, \mathtt{r}, o')$ are sampled for each test triple $t = (s, \mathtt{r}, o)$ to be predicted. Of these negatives, the triples already existing in the training, validation or test set are filtered out (Bordes et al., 2013). The test triple is then ranked against the remaining negatives based on the scores predicted by the KGE model. The state-of-art evaluation metrics reported over the entire set are (i) *MR*: mean of the ranks, (ii) *MRR*: mean of the reciprocals of ranks and (iii) *Hits@n*: number of triples ranked in top-n.

3 Poisoning Knowledge Graph Embeddings via Instance Attribution

We consider an adversarial attacker that aims to degrade the KGE model's predictive performance on a set of missing triples that have been ranked highly plausible by the model. We denote these *target triples* as $\mathcal{Z} := \{z := (z_s, z_\mathtt{r}, z_o)\}$. Since the predicted ranks are based on the predicted scores; to reduce the predicted rank of a target triple, we craft perturbations to the training data that aim to reduce the predicted score of the target triple.

Threat Model: We use the same threat model as the state-of-art poisoning attacks on KGE models (Pezeshkpour et al., 2019; Zhang et al., 2019a). We focus on the white-box attack setting where the attacker has full knowledge of the victim model architecture and access to the learned embeddings. However, they cannot perturb the architecture or the embeddings directly; but only through perturbations in the training data. We study both *adversarial additions* and *adversarial deletions*. In both settings, the attacker is restricted to making only one edit in the neighbourhood of the target triple. The neighbourhood of the target triple $z := (z_s, z_\mathtt{r}, z_o)$ is the set of triples that have the same subject or the same object as the target triple, i.e. $\mathcal{X} := \{x := (x_s, x_\mathtt{r}, x_o) \,|\, x_s \in \{z_s, z_o\} \vee x_o \in \{z_s, z_o\}\}$.

3.1 Instance Attribution Methods

For adversarial deletions, we want to identify the training triples that have influenced the KGE model's prediction on the target triple. Deleting these influential triples from the training set will likely degrade the prediction on the target triple. Thus, we define an *influence score* $\phi(z, x) : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ for the pairs of triples $(z, x) \in \mathcal{T} \times \mathcal{T}$ which indicates the influence of training triple x on the prediction of target triple z. Larger values of the influence score $\phi(z, x)$ indicate that removing x from the training data would cause larger reduction in the predicted score on z.

Trivially, we can compute the influence score for a training triple by removing the triple and re-training the KGE model. However, this is a prohibitively expensive step that requires re-training a new KGE model for every candidate influential triple. Thus, we use the following instance-attribution methods from Interpretable Machine Learning (Molnar, 2019) to estimate the influence score $\phi(z, x)$ without re-training the model.

3.1.1 Instance Similarity

We estimate the influence of training triple x on the prediction of target triple z based on the similarity of their feature representations. The intuition behind these metrics is to identify the training triples that a KGE model has learnt to be similar to the target triple and thus (might) have influenced the model's prediction on the target triple.

Computing this similarity between triples requires feature vector representations for the triples. We note that while the standard KGE scoring functions assign a scalar score to the triples, this scalar value is obtained by reducing over the embedding dimension. For example, in the tri-linear dot product for DistMult, the embeddings of subject, relation and object are multiplied element-wise and then the scalar score for the triple is obtained by summing over the embedding dimension, i.e. $f_t := \langle \boldsymbol{e}_s, \boldsymbol{e}_\mathtt{r}, \boldsymbol{e}_o \rangle := \sum_{i=1}^{k} \boldsymbol{e}_{s_i} \boldsymbol{e}_{\mathtt{r}_i} \boldsymbol{e}_{o_i}$ where k is the embedding dimension.

Thus, to obtain feature vector representations for the triples $\boldsymbol{f}_t : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}^k$, we use the state-of-art KGE scoring functions without reduction over the embedding dimension. For the DistMult model, the triple feature vector is $\boldsymbol{f} := \boldsymbol{e}_s \circ \boldsymbol{e}_\mathtt{r} \circ \boldsymbol{e}_o$ where $\circ$ is the Hadamard (element-wise) product. Table 1 shows the feature vector scores for different KGE models used in this research.

Given the feature vectors for target triples $\boldsymbol{f}(z)$ and training triples $\boldsymbol{f}(x)$, we follow Hanawa et al. (2021) and define the following metrics.

Dot Metric: This metric computes the similarity between target and training instances as the dot product of their feature vectors. That is, $\phi_{dot}(z, x) := \langle \boldsymbol{f}(z), \boldsymbol{f}(x) \rangle$

$\boldsymbol{\ell_2}$ Metric: This metric computes similarity as the negative Euclidean distance between the feature vectors of target instance and test instance. That is, $\phi_{\ell_2}(z, x) := - \|\boldsymbol{f}(z) - \boldsymbol{f}(x)\|_2$

Cosine Metric: This metric computes similarity as the dot product between $\boldsymbol{\ell_2}$ normalized feature vectors of target and test instance, i.e. it ignores the magnitude of the vectors and only relies on the angle between them. That is, $\phi_{cos}(z, x) := \cos(\boldsymbol{f}(z), \boldsymbol{f}(x))$

Here, we denote the dot product for two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ as $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \sum_{i=1}^{p} a_i b_i$; the $\boldsymbol{\ell_2}$ norm of a vector as $\|\boldsymbol{a}\|_2 := \sqrt{\langle \boldsymbol{a}, \boldsymbol{a} \rangle}$; and the cos similarity between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ as $\cos(\boldsymbol{a}, \boldsymbol{b}) := {}^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}/_{\|\boldsymbol{a}\|_2 \|\boldsymbol{b}\|_2}$.

3.1.2 Gradient Similarity

We represent the gradient of the loss for triple z w.r.t. model parameters as $\boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}) := \nabla_{\boldsymbol{\theta}} \mathcal{L}(z, \widehat{\boldsymbol{\theta}})$. Gradient similarity metrics compute similarity between the gradients due to target triple z and the gradients due to training triple x. The intuition is to assign higher influence to training triples that have similar effect on the model's parameters as the target triple; and are therefore likely to impact the prediction on target triple (Charpiat et al., 2019). Thus, using the same similarity functions as Instance Similarity metrics, we define the following three metrics for gradient similarity - Gradient Dot (GD), Gradient $\boldsymbol{\ell_2}$ (GL) and Gradient Cosine (GC).

GD(dot): $\phi_{\text{GD}}(z, x) := \langle \boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}), \boldsymbol{g}(x, \widehat{\boldsymbol{\theta}}) \rangle$

GL ($\boldsymbol{\ell_2}$): $\phi_{\text{GL}}(z, x) := - \left\| \boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}) - \boldsymbol{g}(x, \widehat{\boldsymbol{\theta}}) \right\|_2$

GC(cos): $\phi_{\text{GC}}(z, x) := \cos(\boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}), \boldsymbol{g}(x, \widehat{\boldsymbol{\theta}}))$

3.1.3 Influence Functions

Influence Functions (IF) is a classic technique from robust statistics and was introduced to explain the predictions of black-box models in Koh and Liang (2017). To estimate the effect of a training point on a model's predictions, it first approximates the effect of removing the training point on the learned model parameters. To do this, it performs a first order Taylor expansion around the learned parameters $\widehat{\boldsymbol{\theta}}$ at the optimality conditions.

Following the derivation in Koh and Liang (2017), the the effect of removing the training triple x on $\widehat{\boldsymbol{\theta}}$ is given by ${}^{d\widehat{\boldsymbol{\theta}}}/_{d\epsilon_i} = \boldsymbol{H}_{\widehat{\boldsymbol{\theta}}}^{-1} \boldsymbol{g}(x, \widehat{\boldsymbol{\theta}})$. Here, $\boldsymbol{H}_{\widehat{\boldsymbol{\theta}}}$ denotes the Hessian of the loss function $\boldsymbol{H}_{\widehat{\boldsymbol{\theta}}} := {}^1/_n \sum_{t \in \mathcal{T}} \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(t, \widehat{\boldsymbol{\theta}})$. Using the chain rule then, we approximate the influence of removing x on the model's prediction at z as $\langle \boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}), {}^{d\widehat{\boldsymbol{\theta}}}/_{d\epsilon_i} \rangle$. Thus, the influence score using IF is defined as:

IF: $\phi_{\text{IF}}(z, x) := \langle \boldsymbol{g}(z, \widehat{\boldsymbol{\theta}}), \boldsymbol{H}_{\widehat{\boldsymbol{\theta}}}^{-1} \boldsymbol{g}(x, \widehat{\boldsymbol{\theta}}) \rangle$

Computing the IF for KGE models poses two challenges - (i) storing and inverting the Hessian matrix is computationally too expensive for a large number of parameters; (ii) the Hessian is not guaranteed to be positive definite and thus, invertible because KGE models are non-convex models. To address both these challenges, we follow the guidelines in Koh and Liang (2017). Instead of computing the exact Hessian matrix, we estimate the Hessian-vector product (HVP) with target triple's gradient. That is, for every target triple z, we precompute the value $\boldsymbol{H}_{\widehat{\boldsymbol{\theta}}}^{-1} \boldsymbol{g}(z, \widehat{\boldsymbol{\theta}})$. Then, for each

neighbourhood triple x in the training set, we compute $\phi_{\text{IF}}(z, x)$ using the pre-computed HVP. Furthermore, we use the stochastic estimator LiSSA (Agarwal et al., 2017) that computes the HVP in linear time using samples from training data. For the second issue of non-convexity, we add a "damping" term to the Hessian so that it is positive definite and invertible. This term is a hyperparameter that is tuned to ensure that all eigenvalues of the Hessian matrix are positive, i.e. the Hessian matrix is positive definite. Further discussion on the validity of Influence Functions for non-convex settings is available in Koh and Liang (2017).

3.2 Adversarial Additions

In this attack setting, the adversarial attacker can only *add* triples to the neighbourhood of target triple. Using the Instance Attribution metrics above, we select the training triple $x := (x_s, x_\mathbf{r}, x_o)$ in the neighbourhood of the target triple $z := (z_s, z_\mathbf{r}, z_o)$ that is most influential to the prediction of z. For brevity, lets assume $x_s = z_s$, i.e. the influential and target triples have the same subject. To generate adversarial addition using the influential triple, we propose to replace x_o with the most dissimilar entity $x_{o'}$. Since the adversarial triple $x' := (x_s, x_\mathbf{r}, x_{o'})$ has the same subject and relation as the influential triple but a different object, it should reduce the influence of the influential triple on the target triple's prediction. This in turn should degrade the model prediction on target triple. For multiplicative models, we select the dissimilar entity $x_{o'}$ using the cosine similarity between x_o and the entities $\mathcal{E}$. For additive models, we use the $\boldsymbol{\ell_2}$ similarity between x_o and the entities $\mathcal{E}$.

4 Evaluation

We evaluate the effectiveness of the proposed attack strategies in degrading the KGE model's predictions on target triples at test time. We follow the state-of-art protocol to evaluate poisoning attacks (Xu et al., 2020) - we train a victim KGE model on the original dataset; generate adversarial deletions or additions using one of the attacks; perturb the original dataset; and train a new KGE model on the perturbed dataset. The hyperparameters for victim and poisoned KGE models are same.

We evaluate our attacks on four state-of-art KGE models - DistMult, ComplEx, ConvE and TransE on two publicly available[1] benchmark datasets - WN18RR and FB15k-237. To be able to evaluate the effectiveness of attacks in *degrading* the predictive performance, we select a subset of the benchmark test triples that has been ranked *highest* (ranks=1) by the victim KGE model. From this subset, we randomly sample 100 triples as the *target triples*. This is to avoid the expensive Hessian inverse estimation in the IF metric for a large number of target triples (for each target triple, this estimation requires one training epoch).

The source code implementation of our experiments is available at `https://github.com/PeruBhardwaj/AttributionAttack`.

Baselines: We evaluate our attacks against baseline methods based on random edits and the state-of-art poisoning attacks. *Random_n* adds or removes a random triple from the neighbourhood of the target triple. *Random_g* adds or removes a random triple globally and is not restricted to the target's neighbourhood. *Direct-Del* and *Direct-Add* are the adversarial deletion and addition attacks proposed in Zhang et al. (2019a). *CRIAGE* is the poisoning attack from Pezeshkpour et al. (2019) and is a baseline for both deletions and additions. *GR* (Gradient Rollback) (Lawrence et al., 2021) uses influence estimation to provide post-hoc explanations for KGE models and can also be used to generate adversarial deletions. Thus, we include this method as a baseline for adversarial deletions.

The attack evaluations in Zhang et al. (2019a); Pezeshkpour et al. (2019); Lawrence et al. (2021) differ with respect to the definition of their *neighbourhood*. Thus, to ensure fair evaluation, we implement all methods with the same neighbourhood - triples that are linked to the subject or object of the target triple (Section 3). We use the publicly available implementations for CRIAGE[2] and Gradient Rollback[3] and implement Direct-Del and Direct-Add ourselves. Further details on datasets, implementation of KGE models, baselines and computing resources is available in Appendix A and B.

Results: For WN18RR and FB15k-237 respectively, Tables 2 and 3 show the degradation in MRR and Hits@1 due to adversarial deletions; and Tables 4 and 5 due to adversarial additions for state-of-art KGE models. Below we discuss different patterns in these results. We also discuss runtime efficiency of the attack methods in Appendix C.1.

[1] https://github.com/TimDettmers/ConvE

[2] https://github.com/pouyapez/criage

[3] https://github.com/carolinlawrence/gradient-rollback

		DistMult		ComplEx		ConvE		TransE	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
Original		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Baseline Attacks	Random_n	0.87 (-13%)	0.82	0.85 (-15%)	0.80	0.82 (-18%)	0.79	0.82 (-18%)	0.70
	Random_g	0.97	0.95	0.96	0.93	0.99	0.98	0.93	0.87
	Direct-Del	0.88	0.77	0.86 (-14%)	0.77	0.71 (-29%)	0.64	0.54 **(-46%)**	0.37
	CRIAGE	0.73 (-27%)	0.66	-	-	Er	Er	-	-
	GR	0.95	0.90	0.93	0.86	0.95	0.91	0.84	0.77
Proposed Attacks	Dot Metric	0.89	0.82	0.85	0.79	0.84 (-16%)	0.80	0.77	0.60
	ℓ_2 Metric	**0.25 (-75%)**	0.16	0.29 (-71%)	0.20	0.88	0.78	0.62	0.50
	Cos Metric	**0.25 (-75%)**	0.16	0.29 (-71%)	0.20	0.87	0.76	0.56 (-44%)	0.40
	GD (dot)	0.28 (-72%)	0.19	0.29	0.21	0.25	0.21	0.71 (-29%)	0.57
	GL (ℓ_2)	0.30	0.20	0.28 **(-72%)**	0.19	0.17 **(-83%)**	0.12	0.72	0.60
	GC (cos)	0.29	0.19	0.29	0.21	0.20	0.16	0.71 (-29%)	0.57
	IF	0.28 (-72%)	0.19	0.29 (-71%)	0.20	0.22 (-78%)	0.17	0.71 (-29%)	0.57

Table 2: Reduction in MRR and Hits@1 due to **adversarial deletions on target triples in WN18RR**. Lower values indicate better results; best results for each model are in bold. First block of rows are the baseline attacks with random edits; second block is state-of-art attacks; remaining are the proposed attacks. For each block, we report the *best* reduction in percentage relative to the original MRR; computed as $(poisoned - original)/original * 100$.

4.1 Comparison with Baselines

We observe that the proposed strategies for adversarial deletions and adversarial additions successfully degrade the predictive performance of KGE models. On the other hand, the state-of-art attacks are ineffective or only partially effective. Adversarial deletions from Gradient Rollback perform similar to random baselines; likely because this method estimates the influence of a training triple as the sum of its gradients over the training process. In this way, it does not account for the target triple in the influence estimation. The method is also likely to be effective only for a KGE model that is trained with a batch size of 1 because it needs to track the gradient updates for each triple.

The CRIAGE baseline is only applicable to DistMult and ConvE. But we found that the method ran into `numpy.linalg.LinAlgError: Singular matrix` error for ConvE; because the Hessian matrix computed from the victim model embeddings was non-invertible[4]. For adversarial deletions on DistMult, the baseline works better than random edits but not the proposed attacks [5]. It is also ineffective against adversarial additions.

We see that Direct-Del is effective on TransE, but not on multiplicative models. This is likely because it estimates the influence of a candidate triple as the *difference* in the triple's score when the neighbour entity embedding is perturbed. The additive nature of this influence score might make it more suitable for additive models. We also see that Direct-Add works similar to random additions, likely because it uses random down-sampling.

The proposed attacks based on instance attribution methods consistently outperform random baselines for adversarial additions and deletions. One exception to this pattern are adversarial additions against TransE on WN18RR. In this case, no influence metric performs better than random neighbourhood edits, though they are all effective for adversarial deletions. One possible reason is that the TransE model is designed to learn hierarchical relations like `_has_part`. We found that the target triples ranked highest by the model have such hierarchical relations; and the influential triple for them has the same relation. That is, the triple $(s_1, \texttt{\_has\_part}, s)$ is the influential triple for $(s, \texttt{\_has\_part}, o)$. Removing this influential triple breaks the hierarchical link between s_1 and s; and degrades TransE predictions on the target. But adding the triple $(s_2, \texttt{\_has\_part}, s)$ still preserves the hierarchical structure which TransE can use to score the target correctly. We provide more examples of such relations in Appendix C.3.

4.2 Comparison across Influence Metrics

We see that the IF and Gradient Similarity metrics show similar degradation in predictive performance.

[4]This issue might be resolved by changing the hyperparameters of the victim KGE model so that the Hessian matrix from the victim embeddings is invertible. But there is no strategic way to make such changes.

[5]Since the influence estimation in CRIAGE uses BCE loss, we also compare for DistMult trained with BCE in Appendix C.2, but the results are similar.

This indicates that the computationally expensive Hessian inverse in the IF can be avoided and simpler metrics can identify influential triples with comparable effectiveness. Furthermore, cos and ℓ_2 based Instance Similarity metrics outperform all other methods for adversarial deletions on DistMult, ComplEx and TransE. This effectiveness of naive metrics indicates the high vulnerability of shallow KGE architectures to data poisoning attacks in practice. In contrast to this, the Input Similarity metrics are less effective in poisoning ConvE, especially significantly on WN18RR. This is likely because the triple feature vectors for ConvE are based on the output from a deeper neural architecture than the Embedding layer alone. Within Instance Similarity metrics, we see that the dot metric is not as effective as others. This could be because the dot product does not normalize the triple feature vectors. Thus, training triples with large norms are prioritized over relevant influential triples (Hanawa et al., 2021).

4.3 Comparison of datasets

We note that the degradation in predictive performance is more significant on WN18RR than on FB15k-237. This is likely due to the sparser graph structure of WN18RR, i.e. there are fewer neighbours per target triple in WN18RR than in FB15k-237 (Appendix C.4). Thus, the model learns its predictions from few influential triples in WN18RR; and removing only one neighbour significantly degrades the model's predictions on the target triple.

On the other hand, because of more neighbours in FB15k-237, the model predictions are likely influenced by a *group* of training triples. Such group effect of training instances on model parameters has been studied in Koh et al. (2019); Basu et al. (2020). We will investigate these methods for *KGE models* on FB15k-237 in the future.

5 Related Work

Cai et al. (2018) and Nickel et al. (2015) provide a comprehensive survey of KGE models. We use the most popular models DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018) and TransE (Bordes et al., 2013).

Our work is most closely related to CRIAGE (Pezeshkpour et al., 2019) and Direct Attack (Zhang et al., 2019a), that study both adversarial additions and deletions against KGE models. But CRIAGE is only applicable to multiplicative models and our experiments (Section 4) show that Direct Attack is effective (with respect to random baselines) on additive models only. On the other hand, our instance attribution methods work for all KGE models. Recently, Lawrence et al. (2021) propose Gradient Rollback to estimate the influence of training triples on the KGE model predictions. The original study uses the influential triples for post-hoc explanations, but they can also be used for adversarial deletions. However, the attack stores the model parameter updates for *all* training triples which are in the order of millions for benchmark datasets; and our experiments (Section 4) show that it performs similar to random deletions. Whereas, our influence estimation methods do not require additional storage and are consistently better than random baselines on all KGE models.

We also study data poisoning attacks against KGE models in Bhardwaj et al. (2021). Here, we exploit the inductive abilities of KGE models to select adversarial additions that improve the predictive performance of the model on a set of decoy triples; which in turn degrades the performance on target triples. These inference patterns based attacks cannot be used for adversarial deletions, but we will perform detailed comparison for adversarial additions in future. In parallel work, Banerjee et al. (2021) study risk aware adversarial attacks with the aim of reducing the exposure risk of an adversarial attack instead of improving the attack effectiveness. Also, previous studies by Minervini et al. (2017) and Cai and Wang (2018) use adversarial regularization on the training loss of KGE models to improve predictive performance. But these adversarial samples are not in the input domain and aim to improve instead of degrade model performance. Poisoning attacks have also been studied against models for undirected and single relational graph data (Zügner et al., 2018; Dai et al., 2018; Xu et al., 2020). But they cannot be applied directly to KGE models because they require gradients of a dense adjacency matrix.

Other related work towards understanding KGE models are Zhang et al. (2019b) and Nandwani et al. (2020) that generate post-hoc explanations in the form of sub-graphs. Also, Trouillon et al. (2019) study the inductive abilities of KGE models as binary relation properties for controlled inference tasks with synthetic datasets. Recently, Allen et al. (2021) interpret the structure of KGE by drawing comparison with word embeddings.

		DistMult		ComplEx		ConvE		TransE	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
Original		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Baseline Attacks	Random_n	0.66 (-34%)	0.52	0.65 (-35%)	0.51	0.62 (-38%)	0.46	0.71 (-29%)	0.56
	Random_g	0.68	0.53	0.65 (-35%)	0.51	0.63	0.50	0.75	0.61
	Direct-Del	0.59 (-41%)	0.42	0.62 (-38%)	0.47	0.57 (-43%)	0.41	0.62 **(-38%)**	0.45
	CRIAGE	0.62	0.47	-	-	Er	Er	-	-
	GR	0.68	0.55	0.66	0.51	0.62	0.45	0.68	0.53
Proposed Attacks	Dot Metric	0.63	0.47	0.64	0.49	0.60	0.44	0.74	0.62
	ℓ_2 Metric	0.58	0.41	0.56 **(-44%)**	0.40	0.53 **(-47%)**	0.35	0.63 (-37%)	0.46
	Cos Metric	0.56 **(-44%)**	0.39	0.57	0.40	0.55	0.38	0.63 (-37%)	0.45
	GD (dot)	0.60	0.44	0.60	0.45	0.55 (-45%)	0.37	0.65	0.49
	GL (ℓ_2)	0.62	0.45	0.60	0.45	0.56	0.41	0.70	0.58
	GC (cos)	0.58 (-42%)	0.42	0.57 (-43%)	0.39	0.57	0.40	0.64 (-36%)	0.48
	IF	0.60 (-40%)	0.44	0.60 (-40%)	0.45	0.58 (-42%)	0.43	0.66 (-34%)	0.52

Table 3: Reduction in MRR and Hits@1 due to **adversarial deletions on target triples in FB15k-237**. Lower values indicate better results. First block of rows are the baseline attacks with random edits; second block is state-of-art attacks; remaining are the proposed attacks. For each block, we report the *best* reduction in percentage relative to the original MRR.

		DistMult		ComplEx		ConvE		TransE	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
Original		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Baseline Attacks	Random_n	0.99 (-1%)	0.98	0.97 (-3%)	0.94	0.99 (-1%)	0.98	0.76 **(-24%)**	0.57
	Random_g	0.99 (-1%)	0.97	0.97 (-3%)	0.95	0.99 (-1%)	0.98	0.93	0.87
	Direct-Add	0.98 (-2%)	0.96	0.95 (-5%)	0.92	0.99 (-1%)	0.98	0.81 (-19%)	0.67
	CRIAGE	0.98 (-2%)	0.97	-	-	Er	Er	-	-
Proposed Attacks	Dot Metric	0.97	0.93	0.95	0.90	0.95 (-5%)	0.91	0.95	0.90
	ℓ_2 Metric	0.89 **(-11%)**	0.78	0.88	0.77	0.98	0.96	0.87 (-13%)	0.83
	Cos Metric	0.89 **(-11%)**	0.78	0.87 (-13%)	0.77	0.99	0.98	0.87 (-13%)	0.83
	GD (dot)	0.90	0.79	0.89	0.79	0.92	0.85	0.80 (-20%)	0.73
	GL (ℓ_2)	0.89 **(-11%)**	0.79	0.86 **(-14%)**	0.73	0.88 **(-12%)**	0.77	0.89	0.83
	GC (cos)	0.90	0.80	0.87	0.76	0.91	0.82	0.80 (-20%)	0.73
	IF	0.90 (-10%)	0.79	0.89 (-11%)	0.79	0.91 (-8.9%)	0.82	0.77 (-23%)	0.67

Table 4: Reduction in MRR and Hits@1 due to **adversarial additions on target triples in WN18RR**. Lower values indicate better results. First block of rows are the baseline attacks with random edits; second block is state-of-art attacks; remaining are the proposed attacks. For each block, we report the *best* reduction in percentage relative to the original MRR.

		DistMult		ComplEx		ConvE		TransE	
		MRR	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**	**MRR**	**Hits@1**
Original		1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Baseline Attacks	Random_n	0.65 (-34%)	0.50	0.69	0.57	0.61 (-39%)	0.46	0.74	0.62
	Random_g	0.66	0.52	0.66 (-34%)	0.52	0.63	0.50	0.73 (-27%)	0.61
	Direct-Add	0.64 (-36%)	0.48	0.66 (-34%)	0.52	0.60 (-40%)	0.45	0.72 (-28%)	0.59
	CRIAGE	0.66	0.50	-	-	Er	Er	-	-
Proposed Attacks	Dot Metric	0.67	0.54	0.65	0.50	0.61	0.46	0.74 (-26%)	0.62
	ℓ_2 Metric	0.64	0.50	0.66	0.52	0.59 (-41%)	0.43	0.74 (-26%)	0.62
	Cos Metric	0.63 (-37%)	0.49	0.63 (-37%)	0.47	0.60	0.43	0.74 (-26%)	0.61
	GD (dot)	0.61 **(-39%)**	0.45	0.65	0.50	0.62	0.46	0.71 **(-29%)**	0.58
	GL (ℓ_2)	0.63	0.48	0.67	0.53	0.61 (-39%)	0.45	0.74	0.60
	GC (cos)	0.62	0.46	0.64 **(-36%)**	0.49	0.61 (-39%)	0.45	0.71 **(-29%)**	0.56
	IF	0.61 **(-39%)**	0.45	0.65 (-35%)	0.50	0.58 **(-42%)**	0.42	0.71 **(-29%)**	0.58

Table 5: Reduction in MRR and Hits@1 due to **adversarial additions on target triples in FB15k-237**. Lower values indicate better results; best results for each model are in bold. First block of rows are the baseline attacks with random edits; second block is state-of-art attacks; remaining are the proposed attacks. For each block, we report the *best* reduction in percentage relative to the original MRR; computed as $(poisoned - original)/original * 100$.

The instance attribution methods we use are also used for post-hoc example-based explanations of black-box models (Molnar, 2019). Hanawa et al. (2021); Charpiat et al. (2019); Pruthi et al. (2020) use Instance or Gradient Similarity on image data. Similar to us, Han et al. (2020); Han and Tsvetkov (2020); Pezeshkpour et al. (2021) use different instance attribution methods, but to provide post-hoc explanations on natural language.

6 Conclusion

We propose data poisoning attacks against KGE models using instance attribution methods and demonstrate that the proposed attacks outperform the state-of-art attacks. We observe that the attacks are particularly effective when the KGE model relies on few training instances to make predictions, i.e. when the input graph is sparse.

We also observe that shallow neural architectures like DistMult, ComplEx and TransE are vulnerable to naive attacks based on Instance Similarity. These models have shown competitive predictive performance by proper hyperparameter tuning (Ruffinelli et al., 2020; Kadlec et al., 2017), making them promising candidates for use in production pipelines. But our research shows that these performance gains can be brittle. This calls for improved KGE model evaluation that accounts for adversarial robustness in addition to predictive performance.

Additionally, as in Bhardwaj (2020); Bhardwaj et al. (2021), we call for future proposals to defend against the security vulnerabilities of KGE models. Some promising directions might be to use adversarial training techniques or train ensembles of models over subsets of training data to prevent the model predictions being influenced by a few triples only. Specification of the model failure modes through adversarial robustness certificates will also improve the usability of KGE models in high-stake domains like healthcare and finance.

Acknowledgements

This research was conducted with the financial support of Accenture Labs and Science Foundation Ireland (SFI) at the ADAPT SFI Research Centre at Trinity College Dublin. The ADAPT SFI Centre for Digital Content Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) through Grant No. 13/RC/2106_P2.

Broader Impact

We study the problem of generating data poisoning attacks against KGE models. These models drive many enterprise products ranging from search engines (Google, Microsoft) to social networks (Facebook) to e-commerce (eBay) (Noy et al., 2019), and are increasingly used in domains with high stakes like healthcare and finance (Hogan et al., 2020; Bendtsen and Petrovski, 2019). Thus, it is important to identify the security vulnerabilities of these models that might be exploited by malicious actors to manipulate the predictions of the model and cause system failure. By highlighting these security vulnerabilities of KGE models, we provide an opportunity to fix them and protect stakeholders from harm. This honours the ACM Code of Ethics to contribute to societal well-being and avoid harm due to computing systems.

Furthermore, to study data poisoning attacks against KGE models, we use the Instance Attribution Methods from Interpretable Machine Learning. These methods can also be used to provide post-hoc explanations for KGE models and thus, improve our understanding of the predictions made by the models. In addition to understanding model predictions, instance based attribution methods can help guide design decisions during KGE model training. There are a vast number of KGE model architectures, training strategies and loss functions, and empirically quantifying the impact of the design choices is often challenging (Ruffinelli et al., 2020). Thus, we would encourage further research on exploring the use of instance attribution methods to understand the impact of these choices on the KGE model predictions. By tracing back the model predictions to the input knowledge graph, we can gain a better understanding of the success or failure of different design choices.

References

Naman Agarwal, Brian Bullins, and Elad Hazan. 2017. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116):1–40.

Carl Allen, Ivana Balazevic, and Timothy Hospedales. 2021. Interpreting knowledge graph relation representation from word embeddings. In *International Conference on Learning Representations*.

Prithu Banerjee, Lingyang Chu, Yong Zhang, Laks V.S. Lakshmanan, and Lanjun Wang. 2021. Stealthy targeted data poisoning attack on knowledge graphs. In

2021 IEEE 37th International Conference on Data Engineering (ICDE), pages 2069–2074. IEEE.

Samyadeep Basu, Xuchen You, and Soheil Feizi. 2020. On second-order group influence functions for black-box predictions. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 715–724. PMLR.

Claus Bendtsen and Slavé Petrovski. 2019. How data and AI are helping unlock the secrets of disease. In *AstraZeneca Blog*.

Peru Bhardwaj. 2020. Towards adversarially robust knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(10):13712–13713.

Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. 2021. Poisoning knowledge graph embeddings via relation inference patterns. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1875–1888, Online. Association for Computational Linguistics.

Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowledge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1470–1480, New Orleans, Louisiana. Association for Computational Linguistics.

Chandrahas, Aditya Sharma, and Partha Talukdar. 2018. Towards understanding the geometry of knowledge graph embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 122–131, Melbourne, Australia. Association for Computational Linguistics.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. 2019. Input similarity from the neural network perspective. In *NeurIPS 2019-33th Annual Conference on Neural Information Processing Systems*.

Luca Costabello, Sumit Pai, Chan Le Van, Rory McGrath, Nicholas McCarthy, and Pedro Tabacof. 2019. AmpliGraph: a Library for Representation Learning on Knowledge Graphs.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1):1811–1818.

Xiaochuang Han and Yulia Tsvetkov. 2020. Fortifying toxic speech detectors against veiled toxicity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7732–7739, Online. Association for Computational Linguistics.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. 2021. Evaluation of similarity-based explanations. In *International Conference on Learning Representations*.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2020. Knowledge graphs. *CoRR*, abs/2003.02320.

Anthony D. Joseph, Blaine Nelson, Benjamin I. P. Rubinstein, and J. D. Tygar. 2019. *Adversarial Machine Learning*. Cambridge University Press.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR.

Carolin Lawrence, Timo Sztyler, and Mathias Niepert. 2021. Explaining neural matrix factorization with gradient rollback. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4987–4995.

Pasquale Minervini, Thomas Demeester, Tim Rocktäschel, and Sebastian Riedel. 2017. Adversarial sets for regularising neural link predictors. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press.

Christoph Molnar. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Yatin Nandwani, Ankesh Gupta, Aman Agrawal, Mayank Singh Chauhan, Parag Singla, and Mausam. 2020. OxKBC: Outcome explanation for factorization based knowledge base completion. In *Automated Knowledge Base Construction*.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43.

Pouya Pezeshkpour, Sarthak Jain, Byron Wallace, and Sameer Singh. 2021. An empirical comparison of instance attribution methods for NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, Online. Association for Computational Linguistics.

Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2019. Investigating robustness and interpretability of link prediction via adversarial modifications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3336–3347, Minneapolis, Minnesota. Association for Computational Linguistics.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Théo Trouillon, Éric Gaussier, Christopher R. Dance, and Guillaume Bouchard. 2019. On inductive abilities of latent factor models for relational learning. *J. Artif. Int. Res.*, 64(1):21–53.

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

Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. 2019a. Data poisoning attack against knowledge graph embedding. In *International Joint Conference on Artificial Intelligence*.

Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019b. Interaction embeddings for prediction and explanation in knowledge graphs. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 96–104.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856.

Appendix

A Dataset Details

We evaluate the proposed attacks on four state-of-art KGE models - DistMult, ComplEx, ConvE and TransE; on two publicly available benchmark datasets for link prediction[6]- WN18RR and FB15k-237. For the KGE model evaluation protocol, we filter out triples from the validation and test set that contain unseen entities.

To assess the attack effectiveness in *degrading* performance on triples predicted as True, we need to select a set of triples that are predicted as True by the victim model. Thus, we select a subset of the benchmark test set that has been ranked the best (i.e. ranks=1) by the victim KGE model. If this subset has more than 100 triples, we randomly sample 100 triples as the *target triples*; otherwise we use all triples as target triples. We do this pre-processing step to avoid the expensive Hessian inverse computation in the Influence Functions (IF) for a large number of target triples - for each target triple, estimating the Hessian inverse (as an HVP) using the LissA algorithm requires one training epoch.

		WN18RR	FB15k-237
Entities		40,559	14,505
Relations		11	237
Training		86,835	272,115
Validation		2,824	17,526
Test		2,924	20,438
Subset with Best Ranks	DistMult	1,109	1,183
	ComplEx	1,198	1,238
	ConvE	1,106	901
	TransE	15	1223

Table 6: Statistics for WN18RR and FB15k-237. We removed triples from the validation and test set that contained unseen entities to ensure that we do not add new entities as adversarial edits. The numbers above (including the number of entities) reflect this filtering.

Table 6 shows the dataset statistics and the number of triples which are ranked best by the different KGE models.

B Training Details

B.1 Training KGE models

We implement four KGE models - DistMult, ComplEx, ConvE and TransE. We use the 1-N training strategy proposed in Lacroix et al. (2018) but we do not add the reciprocal relations. Thus, for each triple, we generate scores for $(s, r) \rightarrow o$ and $(o, r) \rightarrow s$.

[6]https://github.com/TimDettmers/ConvE

	WN18RR		FB15k-237	
	MRR	Hits@1	MRR	Hits@1
DistMult	0.48	0.44	0.34	0.24
ComplEx	0.51	0.47	0.34	0.25
ConvE	0.44	0.41	0.32	0.23
TransE	0.21	0.02	0.33	0.24

Table 7: MRR and Hits@1 results for original KGE models on WN18RR and FB15k-237

For TransE scoring function, we use the L2 norm. The loss function used for all models is Pytorch's `CrossEntropyLoss`. For regularization, we use N3 regularization and input dropout on DistMult and ComplEx; input dropout, hidden dropout and feature dropout on ConvE; and L2 regularization (Bordes et al., 2013) and input dropout for TransE.

We do not use early stopping to ensure same hyperparameters for original and poisoned KGE models. We use an embedding size of 200 for all models on both datasets. An exception is TransE model for WN18RR, where we used embedding dim = 100 due to the expensive time and space complexity of 1-N training for TransE. We manually tuned the hyperparameters for KGE models based on suggestions from state-of-art implementations (Ruffinelli et al., 2020; Dettmers et al., 2018; Lacroix et al., 2018; Costabello et al., 2019).

Table 7 shows the MRR and Hits@1 for the original KGE models on WN18RR and FB15k-237. To re-train the KGE model on poisoned dataset, we use the same hyperparameters as the original model. We run all model training, adversarial attacks and evaluation on a shared HPC cluster with Nvidia RTX 2080ti, Tesla K40 and V100 GPUs.

To ensure reproducibility, our source code is publicly available on GitHub at `https://github.com/PeruBhardwaj/AttributionAttack`. The results in Section 4 can be reproduced by passing the argument `reproduce-results` to the attack scripts. Example commands for this are available in the bash scripts in our codebase. The hyperparameter used to generate the results can be inspected in the `set_hyperparams()` function in the file `utils.py` or in the log files.

For the LissA algorithm used to estimate the Hessian inverse in Influence Functions, we select the hyperparameter values using suggestions from Koh and Liang (2017). The values are selected

to ensure that the Taylor expansion in the estimator converges. These hyperparameter values for our experiments are available in the function `set_if_params()` in the file `utils.py` of the accompanying codebase.

B.2 Baseline Implementation Details

One of the baselines in Section 4 of the main paper is the Direct-Del and Direct-Add attack from (Zhang et al., 2019a). The original study evaluated the method for the neighbourhood of subject of the target triple. We extend it for both subject and object to ensure fair comparison with other attacks. Since no public implementation is available, we implement our own.

	WN18RR		
	Original	**High**	**Low**
DistMult	1.00	0.98	0.98
ComplEx	1.00	0.96	0.95
ConvE	1.00	0.99	0.99
TransE	1.00	0.81	0.86
	FB15k-237		
	Original	**High**	**Low**
DistMult	1.00	0.64	0.64
ComplEx	1.00	0.67	0.66
ConvE	1.00	0.62	0.60
TransE	1.00	0.72	0.73

Table 8: MRR of KGE models trained on original datasets and poisoned datasets from the Direct-Add baseline attack in Zhang et al. (2019a). High, Low indicate the high (20%) and low percentage (5%) of candidates selected from random down-sampling.

The Direct-Add attack is based on computing a perturbation score for all possible candidate additions. Since the search space for candidate additions is of the order $\mathcal{E} \times \mathcal{R}$ (where $\mathcal{E}$ and $\mathcal{R}$ are the set of entities and relations), it uses random down sampling to filter out the candidates. The percent of triples down sampled are not reported in the original paper and a public implementation is not available. So, in this paper, we pick a high and a low value for the percentage of triples to be down-sampled and generate adversarial additions for both fractions. We arbitrarily choose 20% of all candidate additions for high; and 5% of all candidate additions as low.

Thus, we generate two poisoned datasets from the attack - one that used a high number of candidates and another that used a low number of candidates. We train two separate KGE models on these datasets to assess the baseline performance. Table 8 shows the MRR of the original model; and poisoned KGE models from attack with high and low down-sampling percents. The results reported for Direct-Add in Section 4 of the main paper are the better of the two results (which show more degradation in performance) for each combination.

C Further Analysis of Proposed Attacks

C.1 Runtime Analysis

We analyze the runtime efficiency of baseline and proposed attack methods for adversarial deletions. For brevity, we consider the attacks on DistMult model, but the results on other models show similar time scales. Table 9 shows the time taken in seconds to select the influential triples for DistMult model on WN18RR and FB15k-237.

		WN18RR	**FB15k-237**
Baseline Attacks	Random_n	0.024	0.057
	Random_g	0.002	0.002
	Direct-Del	0.407	0.272
	CRIAGE	2.235	75.117
	GR	29.919	174.191
Proposed Attacks	Dot Metric	0.288	0.342
	ℓ_2 Metric	0.057	0.067
	Cos Metric	0.067	0.148
	GD (dot)	7.354	109.015
	GL (ℓ_2)	8.100	120.659
	GC (cos)	9.478	141.276
	IF	4751.987	4750.404

Table 9: Time taken in seconds for baseline and proposed attacks to generate influential triples for DistMult on WN18RR and FB15k-237

We see that the Instance Similarity metrics (dot metric, ℓ_2 metric, cos metric) are more efficient than the state-of-art attacks (Direct-Del, CRIAGE and GR). Furthermore, the ℓ_2 metric is almost as quick as random triple selection. The efficiency of Gradient Similarity metrics is also better than or equivalent to CRIAGE and GR.

Only the attack method based on IF is much slower than any other method. This is because estimating the Hessian inverse in IF requires one training epoch for every target triple, that is, we run 100 training epochs to get the influential triples for 100 target triples. However, our results in Section 4.2 of the main paper show that this expensive computation does not provide improved adversarial deletions, and thus, might be unnecessary to select influential triples for KGE models.

Target Relation	Influential Relation
_has_part	_has_part
_synset_domain_topic_of	_synset_domain_topic_of
_has_part	_has_part
_synset_domain_topic_of	_synset_domain_topic_of
_synset_domain_topic_of	_synset_domain_topic_of
_synset_domain_topic_of	_synset_domain_topic_of
_instance_hypernym	_instance_hypernym
_synset_domain_topic_of	_synset_domain_topic_of
_instance_hypernym	_synset_domain_topic_of
_synset_domain_topic_of	_synset_domain_topic_of
_member_meronym	_derivationally_related_form
_synset_domain_topic_of	_synset_domain_topic_of
_has_part	_has_part
_member_meronym	_member_meronym
_synset_domain_topic_of	_synset_domain_topic_of

Table 10: Relations from the target triples and influential triples (adversarial deletions) for the cos metric on WN18RR-TransE. This combination has 15 target triples and the table shows the relations for all of them.

C.2 Additional Comparison with CRIAGE

The baseline attack method CRIAGE estimates the influence of a training triple using the BCE loss and is thus likely to be effective only for KGE models that are trained with BCE loss. In Section 4.1, we found that the proposed attacks are more effective than the baseline attack.

But since our original models are trained with cross-entropy loss, we perform an additional analysis of the Instance Similarity attacks against CRIAGE for the DistMult model trained with BCE loss. Table 11 shows the reduction in MRR and Hits@1 due to adversarial deletions in this training setting. We find that the Instance Similarity attacks outperform the baseline for this setting as well.

	WN18RR		FB15k-237	
	MRR	**Hits@1**	**MRR**	**Hits@1**
Original	1.00	1.00	1.00	1.00
CRIAGE	0.67	0.63	0.63	0.46
Dot Metric	0.86	0.81	0.61	0.44
ℓ_2 Metric	**0.12**	**0.06**	0.60	0.43
Cos Metric	**0.12**	**0.06**	**0.58**	**0.38**

Table 11: Reduction MRR and Hits@1 due to adversarial deletions for DistMult (trained with BCE loss) on WN18RR and FB15k-237

C.3 Analysis of Instance Attribution Methods on WN18RR-TransE

For the TransE model on WN18RR, we found that the instance attribution methods lead to effective adversarial deletions with respect to random baselines, but not adversarial additions (Section 4.1 of main paper). A possible reason is based on the ability of TransE model hierarchical relations, i.e. the relations that represent a hierarchy between the subject and object entities. For example, $(s, \texttt{\_has\_part}, o)$ indicates that s is the parent node for o in a hierarchy.

We select the Instance Similarity method cos metric for further analysis. It performs the best of all instance attribution methods for adversarial deletions, but performs worse than random neighbourhood edits for adversarial additions. Table 10 shows the relations in the target triples and the influential triples (i.e. adversarial deletions) selected by cos metric.

We see that the target triples contain mostly hierarchical relations like `_synset_domain_topic_of` and `_has_part`. Also the cos metric identifies influential triples with same relations. And since our adversarial additions are only based on modifying the entity in the influential triple, these edits improve the hierarchy structure of the graph instead of breaking it. Thus, these edits perform well for adversarial deletions, but not for additions.

C.4 Neighbourhood Sparsity Comparison on WN18RR and FB15k-237

In Section 4.3 of the main paper, we found that the proposed attacks are significantly more effective for WN18RR than for FB15k-237. This is likely because there are fewer triples in the neighbourhood of target triples for WN18RR than for FB15k-237. The graph in Figure 2 shows the median number of neighbours of the target triples for WN18RR and FB15k-237. We report median (instead of mean) because of large standard deviation in the number of target triple neighbours for FB15k-237.

We see that the target triple's neighbourhood for WN18RR is significantly sparser than the neighbourhood for FB15k-237. Thus, since the KGE model predictions are learned from fewer triples for WN18RR, it is also easier to perturb these results with fewer adversarial edits.

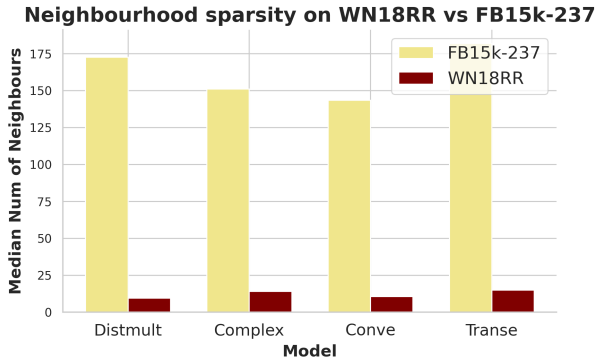

Figure 2: Comparison of the median number of neighbouring triples of target triples from WN18RR and FB15k-237 for DistMult, ComplEx, ConvE and TransE.