Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications

Pouya Pezeshkpour
University of California
Irvine, CA
pezeshkp@uci.edu

Yifan Tian
University of California
Irvine, CA
yifant@uci.edu

Sameer Singh
University of California
Irvine, CA
sameer@uci.edu

Abstract

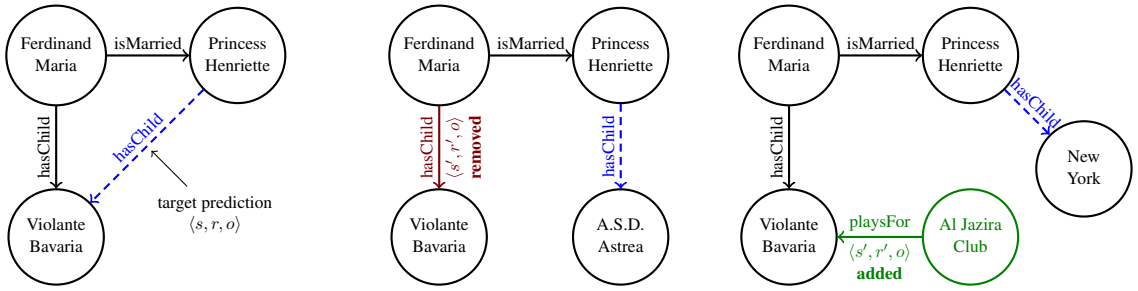
Representing entities and relations in an embedding space is a well-studied approach for machine learning on relational data. Existing approaches, however, primarily focus on improving accuracy and overlook other aspects such as robustness and interpretability. In this paper, we propose adversarial modifications for link prediction models: identifying the fact to add into or remove from the knowledge graph that changes the prediction for a target fact after the model is retrained. Using these single modifications of the graph, we identify the most influential fact for a predicted link and evaluate the sensitivity of the model to the addition of fake facts. We introduce an efficient approach to estimate the effect of such modifications by approximating the change in the embeddings when the knowledge graph changes. To avoid the combinatorial search over all possible facts, we train a network to *decode* embeddings to their corresponding graph components, allowing the use of gradient-based optimization to identify the adversarial modification. We use these techniques to evaluate the robustness of link prediction models (by measuring sensitivity to additional facts), study interpretability through the facts most responsible for predictions (by identifying the most influential neighbors), and detect incorrect facts in the knowledge base.

1 Introduction

Knowledge graphs (KG) play a critical role in many real-world applications such as search, structured data management, recommendations, and question answering. Since KGs often suffer from incompleteness and noise in their facts (links), a number of recent techniques have proposed models that *embed* each entity and relation into a vector space, and use these embeddings to predict facts. These dense representation models for link prediction include tensor factorization [Nickel et al., 2011, Socher et al., 2013, Yang et al., 2015], algebraic operations [Bordes et al., 2011, 2013b, Dasgupta et al., 2018], multiple embeddings [Wang et al., 2014, Lin et al., 2015, Ji et al., 2015, Zhang et al., 2018], and complex neural models [Dettmers et al., 2018, Nguyen et al., 2018]. However, there are only a few studies [Kadlec et al., 2017, Sharma et al., 2018] that investigate the quality of the different KG models. There is a need to go beyond just the accuracy on link prediction, and instead focus on whether these representations are robust and stable, and what facts they make use of for their predictions.

In this paper, our goal is to design approaches that minimally change the graph structure such that the prediction of a target fact changes the most after *the embeddings are relearned*, which we collectively call *Completion Robustness and Interpretability via Adversarial Graph Edits* (CRIAGE). First, we consider perturbations that remove a neighboring link for the target fact, thus identifying the *most influential* related fact, providing an explanation for the model's prediction. As an example, consider the excerpt from a KG in Figure 1a with two observed facts, and a target *predicted fact* that Princes Henriette is the parent of Violante Bavaria. Our proposed graph perturbation, shown in Figure 1b, identifies the existing fact that Ferdinal Maria is the father of Violante Bavaria as the one when removed and model retrained, will change the prediction of Princes Henriette's child. We also study attacks that add a new, fake fact into the KG to evaluate the robustness and sensitivity of link prediction models to small additions to the graph. An example attack for the original graph in Figure 1a, is depicted in Figure 1c. Such perturbations to the the training data are from a family of adversarial modifications that have been applied to other machine learning tasks, known as *poisoning* [Biggio et al., 2012, Corona et al., 2013, Biggio

Proceedings of NAACL-HLT 2019, pages 3336–3347
Minneapolis, Minnesota, June 2 - June 7, 2019.



(a) KG, with the target prediction (b) After removing a fact (c) After adding a fact

Figure 1: **Completion Robustness and Interpretability via Adversarial Graph Edits (CRIAGE):** Change in the graph structure that changes the prediction of the *retrained* model, where (a) is the original sub-graph of the KG, (b) removes a neighboring link of the target, resulting in a change in the prediction, and (c) shows the effect of adding an attack triple on the target. These modifications were identified by our proposed approach.

et al., 2014, Zügner et al., 2018].

Since the setting is quite different from traditional adversarial attacks, search for link prediction adversaries brings up unique challenges. To find these minimal changes for a target link, we need to identify the fact that, when added into or removed from the graph, will have the biggest impact on the predicted score of the target fact. Unfortunately, computing this change in the score is expensive since it involves retraining the model to recompute the embeddings. We propose an efficient estimate of this score change by approximating the change in the embeddings using Taylor expansion. The other challenge in identifying adversarial modifications for link prediction, especially when considering addition of fake facts, is the combinatorial search space over possible facts, which is intractable to enumerate. We introduce an *inverter* of the original embedding model, to *decode* the embeddings to their corresponding graph components, making the search of facts tractable by performing efficient gradient-based continuous optimization.

We evaluate our proposed methods through following experiments. First, on relatively small KGs, we show that our approximations are accurate compared to the true change in the score. Second, we show that our additive attacks can effectively reduce the performance of state of the art models [Yang et al., 2015, Dettmers et al., 2018] up to 27.3% and 50.7% in Hits@1 for two large KGs: WN18 and YAGO3-10. We also explore the utility of adversarial modifications in explaining the model predictions by presenting rule-like descriptions of the most influential neighbors. Finally, we use adversaries to detect errors in the KG, obtaining up to 55% accuracy in detecting errors.

2 Background and Notation

In this section, we briefly introduce some notations, and existing relational embedding approaches that model knowledge graph completion using dense vectors. In KGs, facts are represented using triples of subject, relation, and object, $\langle s, r, o \rangle$, where $s, o \in \xi$, the set of entities, and $r \in \mathcal{R}$, the set of relations. To model the KG, a scoring function $\psi : \xi \times \mathcal{R} \times \xi \rightarrow \mathbb{R}$ is learned to evaluate whether any given fact is true. In this work, we focus on *multiplicative* models of link prediction[1], specifically DistMult [Yang et al., 2015] because of its simplicity and popularity, and ConvE [Dettmers et al., 2018] because of its high accuracy. We can represent the scoring function of such methods as $\psi(s, r, o) = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) \cdot \mathbf{e}_o$, where $\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o \in \mathbb{R}^d$ are embeddings of the subject, relation, and object respectively. In DistMult, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) = \mathbf{e}_s \odot \mathbf{e}_r$, where $\odot$ is element-wise multiplication operator. Similarly, in ConvE, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ is computed by a convolution on the concatenation of $\mathbf{e}_s$ and $\mathbf{e}_r$.

We use the same setup as Dettmers et al. [2018] for training, i.e., incorporate binary cross-entropy loss over the triple scores. In particular, for subject-relation pairs (s, r) in the training data G, we use binary $y_o^{s,r}$ to represent negative and positive facts. Using the model's probability of truth as $\sigma(\psi(s, r, o))$ for $\langle s, r, o \rangle$, the loss is defined as:

$$\mathcal{L}(G) = \sum_{(s,r)} \sum_{o} y_o^{s,r} \log(\sigma(\psi(s, r, o))) + (1 - y_o^{s,r}) \log(1 - \sigma(\psi(s, r, o))). \quad (1)$$

Gradient descent is used to learn the embeddings $\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o$, and the parameters of $\mathbf{f}$, if any.

[1] As opposed to *additive* models, such as TransE [Bordes et al., 2013a], as categorized in Sharma et al. [2018].

3 Completion Robustness and Interpretability via Adversarial Graph Edits (CRIAGE)

For adversarial modifications on KGs, we first define the space of possible modifications. For a target triple $\langle s, r, o\rangle$, we constrain the possible triples that we can remove (or inject) to be in the form of $\langle s', r', o\rangle$ i.e s' and r' may be different from the target, but the object is not. We analyze other forms of modifications such as $\langle s, r', o'\rangle$ and $\langle s, r', o\rangle$ in appendices A.1 and A.2, and leave empirical evaluation of these modifications for future work.

3.1 Removing a fact (CRIAGE-Remove)

For explaining a target prediction, we are interested in identifying the observed fact that has the most influence (according to the model) on the prediction. We define *influence* of an observed fact on the prediction as the change in the prediction score if the observed fact was not present when the embeddings were learned. Previous work have used this concept of influence similarly for several different tasks [Kononenko et al., 2010, Koh and Liang, 2017]. Formally, for the target triple $\langle s, r, o\rangle$ and observed graph G, we want to identify a neighboring triple $\langle s', r', o\rangle \in G$ such that the score $\psi(s, r, o)$ when trained on G and the score $\overline{\psi}(s, r, o)$ when trained on $G - \{\langle s', r', o\rangle\}$ are maximally different, i.e.

$$\underset{(s',r')\in \text{Nei}(o)}{\text{argmax}} \; \Delta_{(s',r')}(s, r, o) \tag{2}$$

where $\Delta_{(s',r')}(s, r, o) = \psi(s, r, o) - \overline{\psi}(s, r, o)$, and $\text{Nei}(o) = \{(s', r') | \langle s', r', o\rangle \in G\}$.

3.2 Adding a new fact (CRIAGE-Add)

We are also interested in investigating the robustness of models, i.e., how sensitive are the predictions to small additions to the knowledge graph. Specifically, for a target prediction $\langle s, r, o\rangle$, we are interested in identifying a single fake fact $\langle s', r', o\rangle$ that, when added to the knowledge graph G, changes the prediction score $\psi(s, r, o)$ the most. Using $\overline{\psi}(s, r, o)$ as the score after training on $G \cup \{\langle s', r', o\rangle\}$, we define the adversary as:

$$\underset{(s',r')}{\text{argmax}} \; \Delta_{(s',r')}(s, r, o) \tag{3}$$

where $\Delta_{(s',r')}(s, r, o) = \psi(s, r, o) - \overline{\psi}(s, r, o)$. The search here is over any possible $s' \in \xi$, which is often in the millions for most real-world KGs, and $r' \in \mathcal{R}$. We also identify adversaries that *increase* the prediction score for specific false triple, i.e., for a target fake fact $\langle s, r, o\rangle$, the adversary is $\text{argmax}_{(s',r')} -\Delta_{(s',r')}(s, r, o)$, where $\Delta_{(s',r')}(s, r, o)$ is defined as before.

3.3 Challenges

There are a number of crucial challenges when conducting such adversarial attack on KGs. First, evaluating the effect of changing the KG on the score of the target fact ($\overline{\psi}(s, r, o)$) is expensive since we need to update the embeddings by retraining the model on the new graph; a very time-consuming process that is at least linear in the size of G. Second, since there are many candidate facts that can be added to the knowledge graph, identifying the most promising adversary through search-based methods is also expensive. Specifically, the search size for unobserved facts is $|\xi| \times |\mathcal{R}|$, which, for example in YAGO3-10 KG, can be as many as $4.5M$ possible facts for a single target prediction.

4 Efficiently Identifying the Modification

In this section, we propose algorithms to address mentioned challenges by (1) approximating the effect of changing the graph on a target prediction, and (2) using continuous optimization for the discrete search over potential modifications.

4.1 First-order Approximation of Influence

We first study the addition of a fact to the graph, and then extend it to cover removal as well. To capture the effect of an adversarial modification on the score of a target triple, we need to study the effect of the change on the vector representations of the target triple. We use $\mathbf{e}_s$, $\mathbf{e}_r$, and $\mathbf{e}_o$ to denote the embeddings of s, r, o at the solution of $\text{argmin}\, \mathcal{L}(G)$, and when considering the adversarial triple $\langle s', r', o\rangle$, we use $\overline{\mathbf{e}_s}$, $\overline{\mathbf{e}_r}$, and $\overline{\mathbf{e}_o}$ for the new embeddings of s, r, o, respectively. Thus $\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r}, \overline{\mathbf{e}_o}$ is a solution to $\text{argmin}\, \mathcal{L}(G \cup \{\langle s', r', o\rangle\})$, which can also be written as $\text{argmin}\, \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$. Similarly, $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ changes to $\mathbf{f}(\overline{\mathbf{e}_s}, \overline{\mathbf{e}_r})$ after retraining.

Since we only consider adversaries in the form of $\langle s', r', o\rangle$, we only consider the effect of the attack on $\mathbf{e}_o$ and neglect its effect on $\mathbf{e}_s$ and $\mathbf{e}_r$. This assumption is reasonable since the adversary is connected with o and directly affects its embedding when added, but it will only have a secondary, negligible effect on $\mathbf{e}_s$ and $\mathbf{e}_r$, in comparison to its

effect on $\mathbf{e}_o$. Further, calculating the effect of the attack on $\mathbf{e}_s$ and $\mathbf{e}_r$ requires a third order derivative of the loss, which is not practical ($O(n^3)$ in the number of parameters). In other words, we assume that $\overline{\mathbf{e}_s} \simeq \mathbf{e}_s$ and $\overline{\mathbf{e}_r} \simeq \mathbf{e}_r$. As a result, to calculate the effect of the attack, $\overline{\psi}(s,r,o) - \psi(s,r,o)$, we need to compute $\overline{\mathbf{e}_o} - \mathbf{e}_o$, followed by:

$$\overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) \quad (4)$$

where $\mathbf{z}_{s,r} = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$. We now derive an efficient computation for $\overline{\mathbf{e}_o} - \mathbf{e}_o$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$ over $\mathbf{e}_o$ is:

$$\nabla_{e_o}\mathcal{L}(\overline{G}) = \nabla_{e_o}\mathcal{L}(G) - (1-\varphi)\mathbf{z}_{s',r'} \quad (5)$$

where $\mathbf{z}_{s',r'} = \mathbf{f}(\mathbf{e}'_s, \mathbf{e}'_r)$, and $\varphi = \sigma(\psi(s',r',o))$. At convergence, after retraining, we expect $\nabla_{e_o}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_o}\mathcal{L}(\overline{G})$ to get:

$$0 \simeq -(1-\varphi)\mathbf{z}^{\intercal}_{s',r'} + (H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})(\overline{\mathbf{e}_o} - \mathbf{e}_o) \quad (6)$$

where H_o is the $d \times d$ Hessian matrix for o, i.e., second order derivative of the loss w.r.t. $\mathbf{e}_o$, computed sparsely. Solving for $\overline{\mathbf{e}_o} - \mathbf{e}_o$ gives us, $\overline{\mathbf{e}_o} - \mathbf{e}_o =:$

$$(1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\intercal}_{s',r'}.$$

In practice, H_o is positive definite, making $H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\overline{\psi}(s,r,o) - \psi(s,r,o) = \mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) \quad (7)$$
$$= \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\intercal}_{s',r'}).$$

Calculating this expression is efficient since H_o is a $d \times d$ matrix (d is the embedding dimension), and $\mathbf{z}_{s,r}, \mathbf{z}_{s',r'} \in \mathbb{R}^d$. Similarly, we estimate the score change of $\langle s,r,o\rangle$ after *removing* $\langle s',r',o\rangle$ as:

$$-\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}^{\intercal}_{s',r'}\mathbf{z}_{s',r'})^{-1}\mathbf{z}^{\intercal}_{s',r'}).$$

4.2 Continuous Optimization for Search

Using the approximations provided in the previous section, Eq. (7) and (4.1), we can use brute force enumeration to find the adversary $\langle s',r',o\rangle$. This approach is feasible when removing an observed triple since the search space of such modifications is usually small; it is the number of observed facts that share the object with the target. On the other hand, finding the most influential unobserved fact to add requires search over a much larger space of all possible unobserved facts (that share the object). Instead, we identify the most influential unobserved fact $\langle s',r',o\rangle$ by using a gradient-based algorithm on vector $\mathbf{z}_{s',r'}$ in the embedding space (reminder, $\mathbf{z}_{s',r'} = \mathbf{f}(\mathbf{e}'_s, \mathbf{e}'_r)$), solving the following continuous optimization problem in $\mathbb{R}^d$:

$$\underset{\mathbf{z}_{s',r'}}{\operatorname{argmax}}\, \Delta_{(s',r')}(s,r,o). \quad (8)$$

After identifying the optimal $\mathbf{z}_{s',r'}$, we still need to generate the pair (s',r'). We design a network, shown in Figure 2, that maps the vector $\mathbf{z}_{s',r'}$ to the entity-relation space, i.e., translating it into (s',r'). In particular, we train an auto-encoder where the encoder is fixed to receive the s and r as one-hot inputs, and calculates $\mathbf{z}_{s,r}$ in the same way as the DistMult and ConvE encoders respectively (using trained embeddings). The decoder is trained to take $\mathbf{z}_{s,r}$ as input and produce s and r, essentially inverting $\mathbf{f}$ and the embedding layers. As our decoder, for DistMult, we pass $\mathbf{z}_{s,r}$ through a linear layer and then use two other linear layers for the subject and the relation separately, providing one-hot vectors as $\tilde{s}$ and $\tilde{r}$. For ConvE, we pass $\mathbf{z}_{s,r}$ through a deconvolutional layer, and then use the same architecture as the DistMult decoder. Although we could use maximum inner-product search [Shrivastava and Li, 2014] for DistMult instead of our defined inverter function, we are looking for a general approach that works across multiple models.

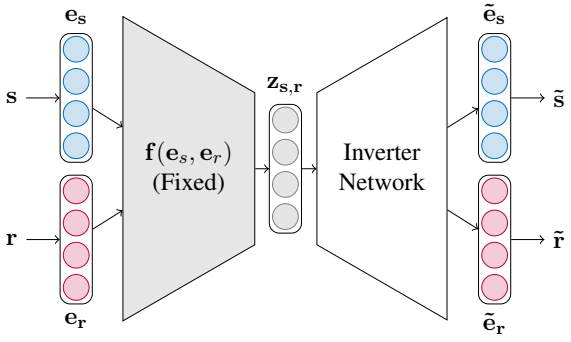


Figure 2: **Inverter Network** The architecture of our inverter function that translate $\mathbf{z}_{s,r}$ to its respective $(\tilde{s}, \tilde{r})$. The encoder component is fixed to be the encoder network of DistMult and ConvE respectively.

We evaluate the performance of our inverter networks (one for each model/dataset) on correctly recovering the pairs of subject and relation from the test set of our benchmarks, given the $\mathbf{z}_{s,r}$. The accuracy of recovered pairs (and of each argument)

	WordNet		YAGO	
	DistMult	ConvE	DistMult	ConvE
Recover s	93.4	96.1	97.2	98.1
Recover r	91.3	95.3	99.0	99.6
Recover $\{s, r\}$	89.5	94.2	96.4	98.0

Table 1: **Inverter Functions Accuracy**, we calculate the accuracy of our inverter networks in correctly recovering the pairs of subject and relation from the test set of our benchmarks.

	# Rels	#Entities	# Train	#Test
Nations	56	14	1592	200
Kinship	26	104	4,006	155
WN18	18	40,943	141,442	5000
YAGO3-10	37	123,170	1,079,040	5000

Table 2: **Data Statistics** of the benchmarks.

is given in Table 1. As shown, our networks achieve a very high accuracy, demonstrating their ability to invert vectors $\mathbf{z}_{s,r}$ to $\{s, r\}$ pairs.

5 Experiment Setup

Datasets To evaluate our method, we conduct several experiments on four widely used KGs. To validate the accuracy of the approximations, we use smaller sized Kinship and Nations KGs for which we can make comparisons against more expensive but less approximate approaches. For the remaining experiments, we use YAGO3-10 and WN18 KGs, which are closer to real-world KGs in their size and characteristics (see Table 2).

Models We implement all methods using the same loss and optimization for training, i.e., AdaGrad and the binary cross-entropy loss. We use validation data to tune the hyperparameters and use a grid search to find the best hyperparameters, such as regularization parameter, and learning rate of the gradient-based method. To capture the effect of our method on link prediction task, we study the change in commonly-used metrics for evaluation in this task: mean reciprocal rank (MRR) and Hits@K. Further, we use the same hyperparameters as in Dettmers et al. [2018] for training link prediction models for these knowledge graphs.

Influence Function We also compare our method with influence function (IF) [Koh and Liang, 2017]. The influence function approximates the effect of upweighting a training sample on the loss for a specific test point. We use IF to approximate the change in the loss after removing a triple as:

$$\mathcal{I}_{\text{up,loss}}(\langle s', r', o\rangle, \langle s, r, o\rangle) = -\nabla_\theta \mathcal{L}(\langle s, r, o\rangle, \hat{\theta})^\intercal H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(\langle s', r', o\rangle, \hat{\theta}) \quad (9)$$

where $\langle s', r', o\rangle$ and $\langle s, r, o\rangle$ are training and test samples respectively, $\hat{\theta}$ represents the optimum parameters and $\mathcal{L}(\langle s, r, o\rangle, \hat{\theta})$ represents the loss function for the test sample $\langle s, r, o\rangle$. Influence function does not scale well, so we only compare our method with IF on the smaller size KGs.

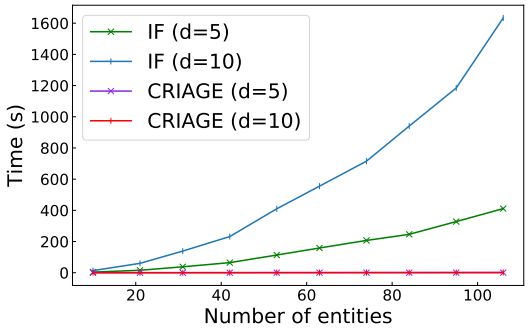


Figure 3: **Influence function vs CRIAGE**. We plot the average time (over 10 facts) of influence function (IF) and CRIAGE to identify an adversary as the number of entities in the Kinship KG is varied (by randomly sampling subgraphs of the KG). Even with small graphs and dimensionality, IF quickly becomes impractical.

6 Experiments

We evaluate CRIAGE by (6.1) comparing CRIAGE estimate with the actual effect of the attacks, (6.2) studying the effect of adversarial attacks on evaluation metrics, (6.3) exploring its application to the interpretability of KG representations, and (6.4) detecting incorrect triples.

6.1 Influence Function vs CRIAGE

To evaluate the quality of our approximations and compare with influence function (IF), we conduct leave one out experiments. In this setup, we take all the neighbors of a random target triple as candidate modifications, remove them one at a time, retrain the model each time, and compute the *exact* change in the score of the target triple. We can use the magnitude of this change in score to rank the candidate triples, and compare this *exact* ranking with ranking as predicted by: CRIAGE-Remove, influence function with and without Hessian matrix, and the original model score (with the intuition that facts that the model is most confident of will have

Methods	Nations				Kinship			
	Adding		Removing		Adding		Removing	
	ρ	τ	ρ	τ	ρ	τ	ρ	τ
Ranking Based on Score	0.03	0.02	-0.01	-0.01	-0.09	-0.06	0.01	0.01
Influence Function without Hessian	0.15	0.12	0.12	0.1	0.77	0.71	0.77	0.71
CRIAGE (Brute Force)	0.95	0.84	0.94	0.85	0.99	0.97	0.99	0.95
Influence Function	0.99	0.95	0.99	0.96	0.99	0.98	0.99	0.98

Table 3: **Ranking modifications by their impact on the target**. We compare the *true* ranking of candidate triples with a number of approximations using ranking correlation coefficients. We compare our method with influence function (IF) with and without Hessian, and ranking the candidates based on their score, on two KGs ($d = 10$, averaged over 10 random targets). For the sake of brevity, we represent the Spearman's ρ and Kendall's τ rank correlation coefficients simply as ρ and τ.

the largest impact when removed). Similarly, we evaluate CRIAGE-Add by considering 200 random triples that share the object entity with the target sample as candidates, and rank them as above.

The average results of Spearman's ρ and Kendall's τ rank correlation coefficients over 10 random target samples is provided in Table 3. CRIAGE performs comparably to the influence function, confirming that our approximation is accurate. Influence function is slightly more accurate because they use the complete Hessian matrix over all the parameters, while we only approximate the change by calculating the Hessian over $\mathbf{e}_o$. The effect of this difference on scalability is dramatic, constraining IF to very small graphs and small embedding dimensionality ($d \leq 10$) before we run out of memory. In Figure 3, we show the time to compute a single adversary by IF compared to CRIAGE, as we steadily grow the number of entities (randomly chosen subgraphs), averaged over 10 random triples. As it shows, CRIAGE is mostly unaffected by the number of entities while IF increases quadratically. Considering that real-world KGs have tens of thousands of times more entities, making IF unfeasible for them.

6.2 Robustness of Link Prediction Models

Now we evaluate the effectiveness of CRIAGE to successfully *attack* link prediction by adding false facts. The goal here is to identify the attacks for triples in the test data, and measuring their effect on MRR and Hits@ metrics (ranking evaluations) after conducting the attack and retraining the model.

Since this is the first work on adversarial attacks for link prediction, we introduce several baselines to compare against our method. For finding the adversarial fact to add for the target triple $\langle s, r, o \rangle$, we consider two baselines: 1) choosing a random fake fact $\langle s', r', o \rangle$ (**Random Attack**); 2) finding (s', r') by first calculating $\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ and then feeding $-\mathbf{f}(\mathbf{e}_s, \mathbf{e}_r)$ to the decoder of the inverter function (**Opposite Attack**). In addition to CRIAGE-Add, we introduce two other alternatives of our method: (1) **CRIAGE-FT**, that uses CRIAGE to *increase* the score of fake fact over a test triple, i.e., we find the fake fact the model ranks second after the test triple, and identify the adversary for them, and (2) **CRIAGE-Best** that selects between CRIAGE-Add and CRIAGE-FT attacks based on which has a higher estimated change in score.

All-Test The result of the attack on all test facts as targets is provided in the Table 4. CRIAGE-Add outperforms the baselines, demonstrating its ability to effectively attack the KG representations. It seems DistMult is more robust against random attacks, while ConvE is more robust against designed attacks. CRIAGE-FT is more effective than CRIAGE-Add since changing the score of a fake fact is easier than of actual facts; there is no existing evidence to support fake facts. We also see that YAGO3-10 models are more robust than those for WN18. Looking at sample attacks (provided in Appendix A.4), CRIAGE mostly tries to change the *type* of the target object by associating it with a subject and a relation for a different entity type.

Uncertain-Test To better understand the effect of attacks, we consider a subset of test triples that 1) the model predicts correctly, 2) difference between their scores and the negative sample with the highest score is minimum. This "Uncertain-Test" subset contains 100 triples from each of the original

	Models	YAGO3-10				WN18			
		All-Test		Uncertain-Test		All-Test		Uncertain-Test	
		MRR	Hits@1	MRR	Hits@1	MRR	Hits@1	MRR	Hits@1
DistMult	DistMult	0.458	37 (0)	1.0	100 (0)	0.938	93.1 (0)	1.0	100 (0)
	+ Adding Random Attack	0.442	34.9 (-2.1)	0.91	87.6 (-12.4)	0.926	91.1 (-2)	0.929	90.4 (-9.6)
	+ Adding Opposite Attack	0.427	33.2 (-3.8)	0.884	84.1 (-15.9)	0.906	87.3 (-5.8)	0.921	91 (-9)
	+ CRIAGE-Add	0.379	29.1 (-7.9)	0.71	58 (-42)	0.89	86.4 (-6.7)	0.844	81.2 (-18.8)
	+ CRIAGE-FT	0.387	27.7 (-9.3)	0.673	50.5 (-49.5)	0.86	79.2 (-13.9)	0.83	74.5 (-25.5)
	+ CRIAGE-Best	0.372	26.9 (-10.1)	0.658	49.3 (-50.7)	0.838	77.9 (-15.2)	0.814	72.7 (-27.3)
ConvE	ConvE	0.497	41.2 (0)	1.0	100 (0)	0.94	93.3 (0)	1.0	100 (0)
	+ Adding Random Attack	0.474	38.4 (-2.8)	0.889	83 (-17)	0.921	90.1 (-3.2)	0.923	89.7 (-10.3)
	+ Adding Opposite Attack	0.469	38 (-3.2)	0.874	81.9 (-18.1)	0.915	88.9 (-4.4)	0.908	88.1 (-11.9)
	+ CRIAGE-Add	0.454	36.9 (-4.3)	0.738	61.5 (-38.5)	0.897	87.8 (-5.5)	0.895	87.6 (-12.4)
	+ CRIAGE-FT	0.441	33.2 (-8)	0.703	57.4 (-42.6)	0.865	80 (-13.3)	0.874	79.5 (-20.5)
	+ CRIAGE-Best	0.423	31.9 (-9.3)	0.677	54.8 (-45.2)	0.849	79.1 (-14.2)	0.858	78.4 (-21.6)

Table 4: **Robustness of Representation Models**, the effect of adversarial attack on link prediction task. We consider two scenario for the target triples, 1) choosing the whole test dataset as the targets (All-Test) and 2) choosing a subset of test data that models are uncertain about them (Uncertain-Test).

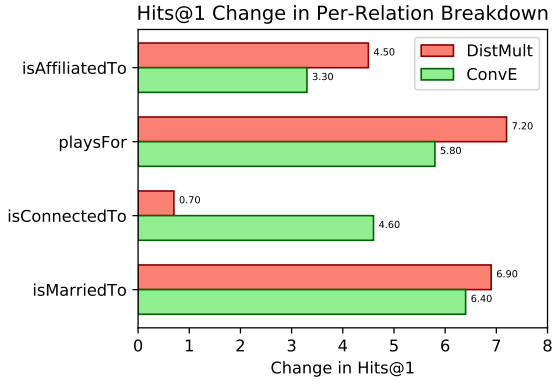


Figure 4: Per-Relation Breakdown showing the effect of CRIAGE-Add on different relations in YAGO3-10.

test sets, and we provide results of attacks on this data in Table 4. The attacks are much more effective in this scenario, causing a considerable drop in the metrics. Further, in addition to CRIAGE significantly outperforming other baselines, they indicate that ConvE's confidence is much more robust.

Relation Breakdown We perform additional analysis on the YAGO3-10 dataset to gain a deeper understanding of the performance of our model. As shown in Figure 4, both DistMult and ConvE provide a more robust representation for isAffiliatedTo and isConnectedTo relations, demonstrating the confidence of models in identifying them. Moreover, the CRIAGE affects DistMult more in playsFor and isMarriedTo relations while affecting ConvE more in isConnectedTo relations.

Rule Body, $R_1(a,c) \wedge R_2(c,b) \Rightarrow$	**Target,** $R(a,b)$
Common to both	
$\text{isConnectedTo}(a,c) \wedge \text{isConnectedTo}(c,b)$	isConnectedTo
$\text{isLocatedIn}(a,c) \wedge \text{isLocatedIn}(c,b)$	isLocatedIn
$\text{isAffiliatedTo}(a,c) \wedge \text{isLocatedIn}(c,b)$	wasBornIn
$\text{isMarriedTo}(a,c) \wedge \text{hasChild}(c,b)$	hasChild
only in DistMult	
$\text{playsFor}(a,c) \wedge \text{isLocatedIn}(c,b)$	wasBornIn
$\text{dealsWith}(a,c) \wedge \text{participatedIn}(c,b)$	participatedIn
$\text{isAffiliatedTo}(a,c) \wedge \text{isLocatedIn}(c,b)$	diedIn
$\text{isLocatedIn}(a,c) \wedge \text{hasCapital}(c,b)$	isLocatedIn
only in ConvE	
$\text{influences}(a,c) \wedge \text{influences}(c,b)$	influences
$\text{isLocatedIn}(a,c) \wedge \text{hasNeighbor}(c,b)$	isLocatedIn
$\text{hasCapital}(a,c) \wedge \text{isLocatedIn}(c,b)$	exports
$\text{hasAdvisor}(a,c) \wedge \text{graduatedFrom}(c,b)$	graduatedFrom
Extractions from DistMult [Yang et al., 2015]	
$\text{isLocatedIn}(a,c) \wedge \text{isLocatedIn}(c,b)$	isLocatedIn
$\text{isAffiliatedTo}(a,c) \wedge \text{isLocatedIn}(c,b)$	wasBornIn
$\text{playsFor}(a,c) \wedge \text{isLocatedIn}(c,b)$	wasBornIn
$\text{isAffiliatedTo}(a,c) \wedge \text{isLocatedIn}(c,b)$	diedIn

Table 5: **Extracted Rules** for identifying the most influential link. We extract the patterns that appear more than 90% times in the neighborhood of the target triple. The output of CRIAGE-Remove is presented in red.

6.3 Interpretability of Models

To be able to understand and interpret why a link is predicted using the opaque, dense embeddings, we need to find out which part of the graph was most influential on the prediction. To provide such explanations for each predictions, we identify the most influential fact using CRIAGE-Remove. Instead of focusing on individual predictions, we aggregate the explanations over the whole dataset for each relation using a simple rule extraction technique: we

Methods	$\langle s', r', o\rangle$ Noise		$\langle s', r, o\rangle$ Noise	
	Hits@1	Hits@2	Hits@1	Hits@2
Random	19.7	39.4	19.7	39.4
Lowest	16	37	26	47
CRIAGE	42	62	55	76

Table 6: **Error Detection Accuracy** in the neighborhood of 100 chosen samples. We choose the neighbor with the least value of $\Delta_{(s',r')}(s, r, o)$ as the incorrect fact. This experiment assumes we know each target fact has exactly one error.

find simple patterns on subgraphs that surround the target triple and the removed fact from CRIAGE-Remove, and appear more than 90% of the time. We only focus on extracting length-2 horn rules, i.e., $R_1(a, c) \wedge R_2(c, b) \Rightarrow R(a, b)$, where $R(a, b)$ is the target and $R_2(c, b)$ is the removed fact.

Table 5 shows extracted YAGO3-10 rules that are common to both models, and ones that are not. The rules show several interesting inferences, such that hasChild is often inferred via married parents, and isLocatedIn via transitivity. There are several differences in how the models reason as well; DistMult often uses the hasCapital as an intermediate step for isLocatedIn, while ConvE *incorrectly* uses isNeighbor. We also compare against rules extracted by Yang et al. [2015] for YAGO3-10 that utilizes the structure of DistMult: they require domain knowledge on types and cannot be applied to ConvE. Interestingly, the extracted rules contain all the rules provided by CRIAGE, demonstrating that CRIAGE can be used to accurately interpret models, including ones that are not interpretable, such as ConvE. These are preliminary steps toward interpretability of link prediction models, and we leave more analysis of interpretability to future work.

6.4 Finding Errors in Knowledge Graphs

Here, we demonstrate another potential use of adversarial modifications: finding erroneous triples in the knowledge graph. Intuitively, if there is an error in the graph, the triple is likely to be inconsistent with its neighborhood, and thus the model should put least trust on this triple. In other words, the error triple should have the least influence on the model's prediction of the training data. Formally, to find the incorrect triple $\langle s', r', o\rangle$ in the neighborhood of the train triple $\langle s, r, o\rangle$, we need to find the triple $\langle s', r', o\rangle$ that results in the *least* change $\Delta_{(s',r')}(s, r, o)$ when removed from the graph.

To evaluate this application, we inject random triples into the graph, and measure the ability of CRIAGE to detect the errors using our optimization. We consider two types of incorrect triples: 1) incorrect triples in the form of $\langle s', r, o\rangle$ where s' is chosen randomly from all of the entities, and 2) incorrect triples in the form of $\langle s', r', o\rangle$ where s' and r' are chosen randomly. We choose 100 random triples from the observed graph, and for each of them, add an incorrect triple (in each of the two scenarios) to its neighborhood. Then, after retraining DistMult on this noisy training data, we identify error triples through a search over the neighbors of the 100 facts. The result of choosing the neighbor with the least influence on the target is provided in the Table 6. When compared with baselines that randomly choose one of the neighbors, or assume that the fact with the lowest score is incorrect, we see that CRIAGE outperforms both of these with a considerable gap, obtaining an accuracy of 42% and 55% in detecting errors.

7 Related Work

Learning relational knowledge representations has been a focus of active research in the past few years, but to the best of our knowledge, this is the first work on conducting adversarial modifications on the link prediction task.

Knowledge graph embedding There is a rich literature on representing knowledge graphs in vector spaces that differ in their scoring functions [Wang et al., 2017, Goyal and Ferrara, 2018, Fooshee et al., 2018]. Although CRIAGE is primarily applicable to multiplicative scoring functions [Nickel et al., 2011, Socher et al., 2013, Yang et al., 2015, Trouillon et al., 2016], these ideas apply to additive scoring functions [Bordes et al., 2013a, Wang et al., 2014, Lin et al., 2015, Nguyen et al., 2016] as well, as we show in Appendix A.3.

Furthermore, there is a growing body of literature that incorporates an extra types of evidence for more informed embeddings such as numerical values [Garcia-Duran and Niepert, 2017], images [Oñoro-Rubio et al., 2017], text [Toutanova et al., 2015, 2016, Tu et al., 2017], and their combinations [Pezeshkpour et al., 2018]. Using CRIAGE, we can gain a deeper understanding of these methods, especially those that build their embeddings wit hmultiplicative scoring functions.

Interpretability and Adversarial Modification There has been a significant recent interest in conducting an adversarial attacks on different machine

learning models [Biggio et al., 2014, Papernot et al., 2016, Dong et al., 2017, Zhao et al., 2018a,b, Brunet et al., 2018] to attain the interpretability, and further, evaluate the robustness of those models. Koh and Liang [2017] uses influence function to provide an approach to understanding black-box models by studying the changes in the loss occurring as a result of changes in the training data. In addition to incorporating their established method on KGs, we derive a novel approach that differs from their procedure in two ways: (1) instead of changes in the loss, we consider the changes in the scoring function, which is more appropriate for KG representations, and (2) in addition to searching for an attack, we introduce a gradient-based method that is much faster, especially for "adding an attack triple" (the size of search space make the influence function method infeasible). Previous work has also considered adversaries for KGs, but as part of training to improve their representation of the graph [Minervini et al., 2017, Cai and Wang, 2018].

Adversarial Attack on KG Although this is the first work on adversarial attacks for link prediction, there are two approaches [Dai et al., 2018, Zügner et al., 2018] that consider the task of adversarial attack on graphs. There are a few fundamental differences from our work: (1) they build their method on top of a path-based representations while we focus on embeddings, (2) they consider node classification as the target of their attacks while we attack link prediction, and (3) they conduct the attack on small graphs due to restricted scalability, while the complexity of our method does not depend on the size of the graph, but only the neighborhood, allowing us to attack real-world graphs.

8 Conclusions

Motivated by the need to analyze the robustness and interpretability of link prediction models, we present a novel approach for conducting adversarial modifications to knowledge graphs. We introduce CRIAGE, completion robustness and interpretability via adversarial graph edits: identifying the fact to add into or remove from the KG that changes the prediction for a target fact. CRIAGE uses (1) an estimate of the score change for any target triple after adding or removing another fact, and (2) a gradient-based algorithm for identifying the most influential modification. We show that CRIAGE can effectively reduce ranking metrics on link prediction models upon applying the attack triples. Further, we incorporate the CRIAGE to study the interpretability of KG representations by summarizing the most influential facts for each relation. Finally, using CRIAGE, we introduce a novel automated error detection method for knowledge graphs. We have release the open-source implementation of our models at: https://pouyapez.github.io/criage.

Acknowledgements

We would like to thank Matt Gardner, Marco Tulio Ribeiro, Zhengli Zhao, Robert L. Logan IV, Dheeru Dua and the anonymous reviewers for their detailed feedback and suggestions. This work is supported in part by Allen Institute for Artificial Intelligence (AI2) and in part by NSF awards #IIS-1817183 and #IIS-1756023. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *International Conference on Machine Learning (ICML)*, 2012.

Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 2014.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI Conference on Artificial Intelligence*, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, 2013a.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems (NeurIPS)*, 2013b.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. Understanding the origins of bias in word embeddings. *arXiv preprint arXiv:1810.03611*, 2018.

Liwei Cai and William Yang Wang. Kbgan: Adversarial learning for knowledge graph embeddings. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

Igino Corona, Giorgio Giacinto, and Fabio Roli. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, 2013.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International Conference on Machine Learning (ICML)*, 2018.

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. *AAAI Conference on Artificial Intelligence*, 2018.

Yinpeng Dong, Hang Su, Jun Zhu, and Fan Bao. Towards interpretable deep neural networks by leveraging adversarial examples. *arXiv preprint arXiv:1708.05493*, 2017.

David Fooshee, Aaron Mood, Eugene Gutman, Mohammadamin Tavakoli, Gregor Urban, Frances Liu, Nancy Huynh, David Van Vranken, and Pierre Baldi. Deep learning for chemical reaction prediction. *Molecular Systems Design & Engineering*, 2018.

Alberto Garcia-Duran and Mathias Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676*, 2017.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 2018.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, 2017.

Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 2010.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI Conference on Artificial Intelligence*, 2015.

P Minervini, T Demeester, T Rocktäschel, and S Riedel. Adversarial sets for regularising neural link predictors. In *Uncertainty in Artificial Intelligence (UAI)*, 2017.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. A novel embedding model for knowledge base completion based on convolutional neural network. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2018.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2016.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *International conference on machine learning (ICML)*, 2011.

Daniel Oñoro-Rubio, Mathias Niepert, Alberto García-Durán, Roberto González-Sánchez, and Roberto J López-Sastre. Representation learning for visual-relational knowledge graphs. *arXiv preprint arXiv:1709.02314*, 2017.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. Embedding multimodal relational data for knowledge base completion. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Aditya Sharma, Partha Talukdar, et al. Towards understanding the geometry of knowledge graph embeddings. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *Neural Information Processing Systems (NeurIPS)*, 2014.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Neural Information Processing Systems (NeurIPS)*, 2013.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. Compositional learning of embeddings for relation paths in knowledge base and text. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, 2016.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. Cane: Context-aware network embedding for relation modeling. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*, 2014.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations (ICLR)*, 2015.

Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. Knowledge graph embedding with hierarchical relation structure. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Mengchen Zhao, Bo An, Yaodong Yu, Sulin Liu, and Sinno Jialin Pan. Data poisoning attacks on multi-task relationship learning. In *AAAI Conference on Artificial Intelligence*, 2018a.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018b.

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

A Appendix

We approximate the change on the score of the target triple upon applying attacks other than the $\langle s', r', o\rangle$ ones. Since each relation appears many times in the training triples, we can assume that applying a single attack will not considerably affect the relations embeddings. As a result, we just need to study the attacks in the form of $\langle s, r', o\rangle$ and $\langle s, r', o'\rangle$. Defining the scoring function as $\psi(s, r, o) = \mathbf{f}(\mathbf{e}_s, \mathbf{e}_r) \cdot \mathbf{e}_o = \mathbf{z}_{s,r} \cdot \mathbf{e}_o$, we further assume that $\psi(s, r, o) = \mathbf{e}_s \cdot \mathbf{g}(\mathbf{e}_r, \mathbf{e}_o) = \mathbf{e}_s \cdot \mathbf{x}_{r,o}$.

A.1 Modifications of the Form $\langle s, r', o'\rangle$

Using similar argument as the attacks in the form of $\langle s', r', o\rangle$, we can calculate the effect of the attack, $\overline{\psi}(s, r, o) - \psi(s, r, o)$ as:

$$\overline{\psi}(s, r, o) - \psi(s, r, o) = (\overline{\mathbf{e}_s} - \mathbf{e}_s)\mathbf{x}_{s,r} \tag{10}$$

where $\mathbf{x}_{s,r} = \mathbf{g}(\mathbf{e}_r, \mathbf{e}_o)$.

We now derive an efficient computation for $(\overline{\mathbf{e}_s} - \mathbf{e}_s)$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s, r', o'\rangle)$ over $\mathbf{e}_s$ is:

$$\nabla_{e_s}\mathcal{L}(\overline{G}) = \nabla_{e_s}\mathcal{L}(G) - (1 - \varphi)\mathbf{x}_{r',o'} \tag{11}$$

where $\mathbf{x}_{r',o'} = \mathbf{g}(\mathbf{e}'_r, \mathbf{e}'_o)$, and $\varphi = \sigma(\psi(s, r', o'))$. At convergence, after retraining, we expect $\nabla_{e_s}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_s}\mathcal{L}(\overline{G})$ to get:

$$\begin{aligned} 0 \simeq &- (1 - \varphi)\mathbf{x}^\intercal_{r',o'} + \\ &(H_s + \varphi(1 - \varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})(\overline{\mathbf{e}_s} - \mathbf{e}_s) \end{aligned} \tag{12}$$

where H_s is the $d \times d$ Hessian matrix for s, i.e. second order derivative of the loss w.r.t. $\mathbf{e}_s$, computed sparsely. Solving for $\overline{\mathbf{e}_s} - \mathbf{e}_s$ gives us:

$$\begin{aligned} &\overline{\mathbf{e}_s} - \mathbf{e}_s = \\ &(1 - \varphi)(H_s + \varphi(1 - \varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^\intercal_{r',o'} \end{aligned}$$

In practice, H_s is positive definite, making $H_s + \varphi(1 - \varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\begin{aligned} &\overline{\psi}(s, r, o) - \psi(s, r, o) = \mathbf{x}_{r,o}(\overline{\mathbf{e}_s} - \mathbf{e}_s) = \\ &((1 - \varphi)(H_s + \varphi(1 - \varphi)\mathbf{x}^\intercal_{r',o'}\mathbf{x}_{r',o'})^{-1}\mathbf{x}^\intercal_{r',o'})\mathbf{x}_{r,o}. \end{aligned} \tag{13}$$

A.2 Modifications of the Form $\langle s, r', o\rangle$

In this section we approximate the effect of attack in the form of $\langle s, r', o\rangle$. In contrast to $\langle s', r', o\rangle$ attacks, for this scenario we need to consider the change in the $\mathbf{e}_s$, upon applying the attack, in approximation of the change in the score as well. Using previous results, we can approximate the $\overline{\mathbf{e}_o} - \mathbf{e}_o$ as:

$$\begin{aligned} &\overline{\mathbf{e}_o} - \mathbf{e}_o = \\ &\qquad (1 - \varphi)(H_o + \varphi(1 - \varphi)\mathbf{z}^\intercal_{s,r'}\mathbf{z}_{s,r'})^{-1}\mathbf{z}^\intercal_{s,r'} \end{aligned} \tag{14}$$

	Target Triple	CRIAGE-Add
DistMult	Brisbane Airport, isConnectedTo, Boulia Airport Jalna District, isLocatedIn, India Quincy Promes, wasBornIn, Amsterdam Princess Henriette, hasChild, Violante Bavaria	Osman Ozköylü, isPoliticianOf, Boulia Airport United States, hasWonPrize, India Gmina Krzeszyce, hasGender, Amsterdam Al Jazira Club, playsFor, Violante Bavaria
ConvE	Brisbane Airport, isConnectedTo, Boulia Airport National Union(Israel), isLocatedIn, Jerusalem Robert Louis, influences, David Leavitt Princess Henriette, hasChild, Violante Bavaria	Victoria Wood, wasBornIn, Boulia Airport Sejad Halilović, isAffiliatedTo, Jerusalem David Louhoungou, hasGender, David Leavitt Jonava, isAffiliatedTo, Violante Bavaria

Table 7: Top adversarial triples for target samples.

and similarly, we can approximate $\overline{\mathbf{e}_s} - \mathbf{e}_s$ as:

$$\overline{\mathbf{e}_s} - \mathbf{e}_s = (1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal \tag{15}$$

where H_s is the Hessian matrix over $\mathbf{e}_s$. Then using these approximations:

$$\mathbf{z}_{s,r}(\overline{\mathbf{e}_o} - \mathbf{e}_o) = \mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^\intercal \mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^\intercal)$$

and:

$$(\overline{\mathbf{e}_s} - \mathbf{e}_s)\mathbf{x}_{r,\bar{o}} = ((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal)\mathbf{x}_{r,\bar{o}}$$

and then calculate the change in the score as:

$$\begin{aligned}&\overline{\psi}(s,r,o) - \psi(s,r,o) = \\ &\mathbf{z}_{s,r}.(\overline{\mathbf{e}_o} - \mathbf{e}_o) + (\overline{\mathbf{e}_s} - \mathbf{e}_s).\mathbf{x}_{r,\bar{o}} = \\ &\mathbf{z}_{s,r}((1-\varphi)(H_o + \varphi(1-\varphi)\mathbf{z}_{s,r'}^\intercal \mathbf{z}_{s,r'})^{-1}\mathbf{z}_{s,r'}^\intercal) + \\ &((1-\varphi)(H_s + \varphi(1-\varphi)\mathbf{x}_{r',o}^\intercal \mathbf{x}_{r',o})^{-1}\mathbf{x}_{r',o}^\intercal)\mathbf{x}_{r,\bar{o}}\end{aligned} \tag{16}$$

A.3 First-order Approximation of the Change For TransE

In here we derive the approximation of the change in the score upon applying an adversarial modification for TransE [Bordes et al., 2013a]. Using similar assumptions and parameters as before, to calculate the effect of the attack, $\overline{\psi}(s,r,o)$ (where $\psi(s,r,o) = |\mathbf{e}_s + \mathbf{e}_r - \mathbf{e}_o|$), we need to compute $\overline{\mathbf{e}_o}$. To do so, we need to derive an efficient computation for $\overline{\mathbf{e}_o}$. First, the derivative of the loss $\mathcal{L}(\overline{G}) = \mathcal{L}(G) + \mathcal{L}(\langle s', r', o\rangle)$ over $\mathbf{e}_o$ is:

$$\nabla_{e_o}\mathcal{L}(\overline{G}) = \nabla_{e_o}\mathcal{L}(G) + (1-\varphi)\frac{\mathbf{z}_{s',r'} - \mathbf{e}_o}{\psi(s',r',o)} \tag{17}$$

where $\mathbf{z}_{s',r'} = \mathbf{e}'_s + \mathbf{e}'_r$, and $\varphi = \sigma(\psi(s',r',o))$. At convergence, after retraining, we expect $\nabla_{e_o}\mathcal{L}(\overline{G}) = 0$. We perform first order Taylor approximation of $\nabla_{e_o}\mathcal{L}(\overline{G})$ to get:

$$0 \simeq (1-\varphi)\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} + (H_o - H_{s',r',o})(\overline{\mathbf{e}_o} - \mathbf{e}_o) \tag{18}$$

$$H_{s',r',o} = (1-\varphi)\varphi\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal(\mathbf{z}_{s',r'} - \mathbf{e}_o)}{\psi(s',r',o)^2} + \frac{1-\varphi}{\psi(s',r',o)} - (1-\varphi)\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal(\mathbf{z}_{s',r'} - \mathbf{e}_o)}{\psi(s',r',o)^3} \tag{19}$$

where H_o is the $d \times d$ Hessian matrix for o, i.e., second order derivative of the loss w.r.t. $\mathbf{e}_o$, computed sparsely. Solving for $\overline{\mathbf{e}_o}$ gives us:

$$\overline{\mathbf{e}_o} = -(1-\varphi)(H_o - H_{s',r',o})^{-1}\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} + \mathbf{e}_o \tag{20}$$

Then, we compute the score change as:

$$\begin{aligned}\overline{\psi}(s,r,o) &= |\mathbf{e}_s + \mathbf{e}_r - \overline{\mathbf{e}_o}| \\ &= |\mathbf{e}_s + \mathbf{e}_r + (1-\varphi)(H_o - H_{s',r',o})^{-1} \\ &\frac{(\mathbf{z}_{s',r'} - \mathbf{e}_o)^\intercal}{\psi(s',r',o)} - \mathbf{e}_o|\end{aligned} \tag{21}$$

Calculating this expression is efficient since H_o is a $d \times d$ matrix.

A.4 Sample Adversarial Attacks

In this section, we provide the output of the CRIAGE-Add for some target triples. Sample adversarial attacks are provided in Table 7. As it shows, CRIAGE-Add attacks mostly try to change the *type* of the target triple's object by associating it with a subject and a relation that require a different entity types.