Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications

Anonymous authors

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 of knowledge base representations, 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 are able to 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., 2015b], 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 call adversarial modifications on link prediction (AMLP). 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’s perturbation, shown in Figure 1b, identifies the existing fact that Ferdinand 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 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.
Investigating Robustness and Interpretability of Link Prediction

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., 2015b, 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 \( \psi : \xi \times \mathcal{R} \times \xi \to \mathbb{R} \) is learned to evaluate whether any given fact is true. In this work, we focus on multiplicative models of link prediction, specifically DistMult [Yang et al., 2015b] 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 \( \psi(s, r, o) = f(e_s, e_r) \cdot e_o \), where \( e_s, e_r, e_o \in \mathbb{R}^d \) are embeddings of the subject, relation, and object respectively. In DistMult, \( f(e_s, e_r) = e_s \odot e_r \), where \( \odot \) is element-wise multiplication operator. Similarly, in ConvE, \( f(e_s, e_r) \) is computed by a convolutional on the concatenation of \( e_s \) and \( e_r \).

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 \( y^{s,r}_o \) to represent negative and positive facts (for observed facts \( y^{s,r}_o = 1 \) and for negative samples \( y^{s,r}_o = 0 \)). Using the model’s probability of truth as \( \sigma(\psi(s, r, o)) \) for \( \langle s, r, o \rangle \), the loss is defined as:

\[
L(G) = \sum_{(s, r) \in G} \sum_o y^{s,r}_o \log(\sigma(\psi(s, r, o))) + (1 - y^{s,r}_o) \log(1 - \sigma(\psi(s, r, o))).
\]

Gradient descent is used to learn the embeddings \( e_s, e_r, e_o \), and the parameters of \( f \), if any.

3. Adversarial Modifications on Link Prediction (AMLP)

For adversarial modifications on KGs, we first define the space of possible modifications. For a target triple \( \langle s, r, o \rangle \), we can remove (or inject) an attack triple in the form of \( \langle s', r', o' \rangle \), where any of \( s', r', \) and \( o' \) are the same as in the target, or are all different. We focus only on \( \langle s', r', o \rangle \) triples as possible changes (we consider other modifications in the appendix).

3.1 Removing a fact (AMLP-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. There were previous works that define the concept of influence in the same way for several different tasks [Kononenko et al., 2010, Koh and Liang, 2017]. Formally, for the target

---

1. As opposed to additive models, such as TransE [Bordes et al., 2013a], as categorized in Sharma et al. [2018].
triple \((s, r, o)\) 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.

\[
\arg\max_{(s', r') \in \text{Nei}(o)} \Delta_{(s', r')} (s, r, o)
\]

where \(\Delta_{(s', r')} (s, r, o) = \psi(s, r, o) - \overline{\psi}(s, r, o)\), and \(\text{Nei}(o) = \{(s', r')|(s', r', o) \in G\}\). We restrict the search area to the neighboring links because they are likely to have the highest influence on the target triple and provide us with a reasonable search space; we leave larger neighborhoods of the observed graph to future work.

### 3.2 Adding a new fact (AMLP-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:

\[
\arg\max_{(s', r')} \Delta_{(s', r')} (s, r, o)
\]

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

\[
\arg\max_{(s', r')} -\Delta_{(s', r')} (s, r, o)
\]

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 \(\psi(s, r, o)\) 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 very 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 the Change

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 $e_s$, $e_r$, and $e_o$ to denote the embeddings of $s, r, o$ at the solution of $\text{argmin } \mathcal{L}(G)$, and when considering the adversarial triple $(s', r', o)$, we use $\mathbf{e}_s, \mathbf{e}_r$, and $\mathbf{e}_o$ for the new embeddings of $s, r, o$, respectively. Thus $\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_o$ is a solution to $\text{argmin } \mathcal{L}(G \cup \{(s', r', o)\})$, which can also be written as $\text{argmin } \mathcal{L}(G) + \mathcal{L}((s', r', o))$. Similarly, $f(e_s, e_r)$ changes to $f(\mathbf{e}_s, \mathbf{e}_r)$ after retraining.

Since we only consider adversaries in the form of $(s', r', o)$, we only consider the effect of the attack on $e_o$ and neglect its effect on $e_s$ and $e_r$. 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 $e_s$ and $e_r$, in comparison to its effect on $e_o$. Further, calculating the effect of the attack on $e_s$ and $e_r$ 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 $\mathbf{e}_s \simeq e_s$ and $\mathbf{e}_r \simeq e_r$. As a result, to calculate the effect of the attack, $\overline{\psi}(s, r, o) - \psi(s, r, o)$, we need to compute $\mathbf{e}_o - e_o$, followed by:

$$\overline{\psi}(s, r, o) - \psi(s, r, o) = z_{s, r}(\mathbf{e}_o - e_o)$$

(4)

where $z_{s, r} = f(e_s, e_r)$. We now derive an efficient computation for $\mathbf{e}_o - e_o$. First, the derivative of the loss $\mathcal{L}(\mathcal{G}) = \mathcal{L}(G) + \mathcal{L}((s', r', o))$ over $e_o$ is:

$$\nabla_{e_o} \mathcal{L}(\mathcal{G}) = \nabla_{e_o} \mathcal{L}(G) - (1 - \varphi)z_{s', r'}$$

(5)

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

$$0 \simeq -(1 - \varphi)z_{s', r'} + (H_o + \varphi(1 - \varphi)z_{s', r'}^T z_{s', r'}) (\mathbf{e}_o - e_o)$$

(6)

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

$$\mathbf{e}_o - e_o = (1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s', r'}^T z_{s', r'})^{-1} z_{s', r'}^T$$

(7)

In practice, $H_o$ is positive definite, making $H_o + \varphi(1 - \varphi)z_{s', r'}^T z_{s', r'}$ positive definite as well, and invertible. Then, we compute the score change as:

$$\overline{\psi}(s, r, o) - \psi(s, r, o) = z_{s, r}((1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s', r'}^T z_{s', r'})^{-1} z_{s', r'}^T)$$

(8)

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

$$-z_{s, r}((1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s', r'}^T z_{s', r'})^{-1} z_{s', r'}^T)$$
4.2 Continuous Optimization for Search

Using the approximations provided in the previous section, Eq. (14) 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 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 \( z_{s',r'} \) in the embedding space (reminder, \( z_{s',r'} = f(e'_s, e'_r) \)), solving the following continuous optimization problem in \( \mathbb{R}^d \):

\[
\arg\max_{z_{s',r'}} \Delta_{\langle s',r' \rangle} (s, r, o). \tag{9}
\]

After identifying the optimal \( z_{s',r'} \), we still need to generate the pair \( (s', r') \). We design a network, shown in Figure 2, that maps the vector \( z_{s',r'} \) 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 \( z_{s,r} \) in the same way as the DistMult and ConvE encoders respectively (using trained embeddings). The decoder is trained to take \( z_{s,r} \) as input and produce \( s \) and \( r \), essentially inverting \( f \) and the embedding layers. As our decoder, for Distmult, we pass \( z_{s,r} \) through a linear layer and then uses 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 \( z_{s,r} \) 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 were looking for a general algorithm which would work across multiple models.

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 \( z_{s,r} \). The accuracy of recovered pairs (and of each argument) is given in Table 1. As shown, our networks achieve a very high accuracy, demonstrating their ability to invert vectors \( z_{s,r} \) to \{s, r\} pairs.
INVESTIGATING ROBUSTNESS AND INTERPRETABILITY OF LINK PREDICTION

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.

<table>
<thead>
<tr>
<th></th>
<th>WordNet</th>
<th>YAGO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DistMult</td>
<td>ConvE</td>
</tr>
<tr>
<td>Correctly Recovering s</td>
<td>93.4</td>
<td>96.1</td>
</tr>
<tr>
<td>Correctly Recovering r</td>
<td>91.3</td>
<td>95.3</td>
</tr>
<tr>
<td>Correctly Recovering {s,r}</td>
<td>89.5</td>
<td>94.2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th># Relations</th>
<th>#Entities</th>
<th># Training Triples</th>
<th># Test Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nations</td>
<td>56</td>
<td>14</td>
<td>1592</td>
<td>200</td>
</tr>
<tr>
<td>Kinship</td>
<td>26</td>
<td>104</td>
<td>4,006</td>
<td>155</td>
</tr>
<tr>
<td>WN18</td>
<td>18</td>
<td>40,943</td>
<td>141,442</td>
<td>5000</td>
</tr>
<tr>
<td>YAGO3-10</td>
<td>37</td>
<td>123,170</td>
<td>1,079,040</td>
<td>5000</td>
</tr>
</tbody>
</table>

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. Table 2 provides the statistics of the datasets.

**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 in the 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:

\[
\mathcal{T}_{up,loss}(⟨s', r', o⟩, ⟨s, r, o⟩) = -\nabla_{\hat{\theta}}\mathcal{L}(⟨s, r, o⟩, \hat{\theta})^\top H^{-1}_{\hat{\theta}} \nabla_{\hat{\theta}}\mathcal{L}(⟨s', r', o⟩, \hat{\theta})
\] (10)

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

**Baselines** Since to the best of our knowledge, this is the first work on the adversarial attack on the link prediction task; we introduce several baselines to compare against our methods. For AMLP-Remove we compare our method with two baselines; 1) randomly removing one of the target triple’s
Figure 3: **Influence function vs AMLP.** We randomly choose a target fact and plot the estimated effect of neighbors (for removing) and 200 random facts (for adding) based on influence function and AMLP, versus the actual effect calculated by leaving/adding each triple and retraining.

Table 3: **Ranking Modifications by Impact on 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 $\rho$ and Kendall’s $\tau$ rank correlation coefficients simply as $\rho$ and $\tau$.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Nations Adding</th>
<th>Nations Removing</th>
<th>Kinship Adding</th>
<th>Kinship Removing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rho$</td>
<td>$\tau$</td>
<td>$\rho$</td>
<td>$\tau$</td>
</tr>
<tr>
<td>Ranking Based on Score</td>
<td>0.03</td>
<td>0.02</td>
<td>-0.01</td>
<td>-0.01</td>
</tr>
<tr>
<td>Influence Function without Hessian</td>
<td>0.15</td>
<td>0.12</td>
<td>0.12</td>
<td>0.1</td>
</tr>
<tr>
<td>AMLP (Brute Force)</td>
<td>0.95</td>
<td>0.84</td>
<td>0.94</td>
<td>0.85</td>
</tr>
<tr>
<td>Influence Function</td>
<td>0.99</td>
<td>0.95</td>
<td>0.99</td>
<td>0.96</td>
</tr>
</tbody>
</table>

neighbors as the attack; 2) choosing the neighbor with nearest $f(e_s', e_r')$ to $f(e_s, e_r)$ as the attack. Along the similar lines, for AMLP-Add to find the attack for target triple $(s, r, o)$ we consider two baselines: 1) choosing a random triple $(s', r', o)$ as our attack; 2) finding $(s', r')$ by first calculate the $f(e_s, e_r)$ and then feed the $-f(e_s, e_r)$ to the decoder component of the trained inverter function.

6. Experiments

We evaluate AMLP by (6.1) comparing AMLP 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 AMLP

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: AMLP-Remove,
INVESTIGATING ROBUSTNESS AND INTERPRETABILITY OF LINK PREDICTION

Figure 4: Influence function vs AMLP. We plot the average time (over 10 facts) of influence function (IF) and AMLP 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.

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 the largest impact when removed). Similarly, we evaluate AMLP-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 $\rho$ and Kendall’s $\tau$ rank correlation coefficients over 10 random target samples is provided in Table 3. AMLP 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 $e_o$. We also show the similarity of the estimates coming from the influence function and AMLP in the Figure 3 when compared to the actual change in the score (computed by retraining), demonstrating that AMLP estimates are fairly accurate even though it is much more efficient than retraining or the influence function. The effect of this difference on scalability is dramatic, limiting IF to very small graphs and small embedding dimensionality ($d \leq 10$) before we run out of memory. In Figure 4, we show the time to compute a single adversary by IF compared to AMLP, as we steadily grow the number of entities (randomly chosen subgraphs), averaged over 10 random triples. As it shows, AMLP 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 infeasible for them.

6.2 Robustness of Link Prediction Models

Now we evaluate the effectiveness of AMLP to successfully attack link prediction task by adding false facts. The goal here is to identify the attacks for triples in the test data, and measuring their
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).

<table>
<thead>
<tr>
<th>Models</th>
<th>YAGO3-10 All-Test</th>
<th>YAGO3-10 Uncertain-Test</th>
<th>WN18 All-Test</th>
<th>WN18 Uncertain-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Hits@1</td>
<td>MRR</td>
<td>Hits@1</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.458</td>
<td>37.0 (0)</td>
<td>0.91</td>
<td>87.6 (-12)</td>
</tr>
<tr>
<td>+ Adding Random Attack</td>
<td>0.442</td>
<td>34.9 (-2.1)</td>
<td>0.879</td>
<td>85.3 (-13.7)</td>
</tr>
<tr>
<td>+ Adding Opposite Attack</td>
<td>0.427</td>
<td>33.2 (-3.8)</td>
<td>0.884</td>
<td>84.1 (-15.9)</td>
</tr>
<tr>
<td>+ Removing Random Attack</td>
<td>0.443</td>
<td>35.3 (-1.7)</td>
<td>0.782</td>
<td>68.0 (-32)</td>
</tr>
<tr>
<td>+ Removing Closest Attack</td>
<td>0.44</td>
<td>34.9 (-2.1)</td>
<td>0.762</td>
<td>64.9 (-35.9)</td>
</tr>
<tr>
<td>+ AMLP-Remove</td>
<td>0.438</td>
<td>34.6 (-2.4)</td>
<td>0.756</td>
<td>63.5 (-36.5)</td>
</tr>
<tr>
<td>+ AMLP-Add</td>
<td>0.379</td>
<td>29.1 (-7.9)</td>
<td>0.71</td>
<td>58.0 (-42)</td>
</tr>
<tr>
<td>+ AMLP-FT</td>
<td>0.387</td>
<td>27.7 (-9.3)</td>
<td>0.673</td>
<td>50.5 (-49.5)</td>
</tr>
<tr>
<td>+ AMLP-Best</td>
<td>0.372</td>
<td><strong>26.9 (-10.1)</strong></td>
<td><strong>0.658</strong></td>
<td><strong>49.3 (-50.7)</strong></td>
</tr>
<tr>
<td>ConvE</td>
<td>0.497</td>
<td>41.2 (0)</td>
<td>1.0</td>
<td>100 (0)</td>
</tr>
<tr>
<td>+ Adding Random Attack</td>
<td>0.474</td>
<td>38.4 (-2.8)</td>
<td>0.889</td>
<td>83.0 (-17)</td>
</tr>
<tr>
<td>+ Adding Opposite Attack</td>
<td>0.469</td>
<td>38.0 (-3.2)</td>
<td>0.874</td>
<td>81.9 (-18.1)</td>
</tr>
<tr>
<td>+ Removing Random Attack</td>
<td>0.47</td>
<td>37.8 (-3.4)</td>
<td>0.821</td>
<td>67.0 (-33)</td>
</tr>
<tr>
<td>+ Removing Closest Attack</td>
<td>0.464</td>
<td>37.9 (-3.3)</td>
<td>0.791</td>
<td>67.2 (-32.8)</td>
</tr>
<tr>
<td>+ AMLP-Remove</td>
<td>0.465</td>
<td>37.4 (-3.8)</td>
<td>0.779</td>
<td>66.2 (-33.8)</td>
</tr>
<tr>
<td>+ AMLP-Add</td>
<td>0.454</td>
<td>36.9 (-4.3)</td>
<td>0.738</td>
<td>61.5 (-38.5)</td>
</tr>
<tr>
<td>+ AMLP-FT</td>
<td><strong>0.441</strong></td>
<td><strong>33.2 (-8)</strong></td>
<td>0.703</td>
<td>57.4 (-42.6)</td>
</tr>
<tr>
<td>+ AMLP-Best</td>
<td><strong>0.423</strong></td>
<td><strong>31.9 (-9.3)</strong></td>
<td><strong>0.677</strong></td>
<td><strong>54.8 (-45.2)</strong></td>
</tr>
</tbody>
</table>

effect on MRR and Hits@ metrics (ranking evaluations) after conducting the attack and retraining the model.

In addition to AMLP-Add and AMLP-Remove, we introduce two other alternatives of our method: (1) AMLP-FT, that uses AMLP 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) AMLP-Best that selects between AMLP-Add and AMLP-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. AMLP-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. AMLP-FT is more effective than AMLP-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 the supplementary materials), AMLP mostly tries to change the type of the target object by associating it with a subject and a relation for a different entity types.

**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 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
INVESTIGATING ROBUSTNESS AND INTERPRETABILITY OF LINK PREDICTION

Figure 5: **Per-Relation Breakdown**, showing effect of AMLP on different relations in YAGO3-10.

Table 5: Top adversarial triples for target samples.

<table>
<thead>
<tr>
<th>Target Triple</th>
<th>AMLP-Add</th>
<th>AMLP-Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>DistMult</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brisbane Airport, isConnectedTo, Boulia Airport</td>
<td>Oman Ozkooli, isPoliticianOf, Boulia Airport</td>
<td>Birdsville Airport, isConnectedTo, Boulia Airport</td>
</tr>
<tr>
<td>Jalna District, isLocatedIn, India</td>
<td>Quincy Promes, wasBornIn, Amsterdam</td>
<td>Maharashtra, isLocatedIn, India</td>
</tr>
<tr>
<td>Princess Henriette, hasChild, Violante Bavaria</td>
<td>Gmina Krzeszowice, hasGender, Amsterdam</td>
<td>A.F.F. Ajax, isLocatedIn, Amsterdam</td>
</tr>
<tr>
<td>ConvE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brisbane Airport, isConnectedTo, Boulia Airport</td>
<td>Victoria Wood, wasBornIn, Boulia Airport</td>
<td>Mount Isa Airport Airport, isConnectedTo, Boulia Airport</td>
</tr>
<tr>
<td>National Union(Israel), isLocatedIn, Jerusalem</td>
<td>Sejad Halilovic, isAffiliatedTo, Jerusalem</td>
<td>Hapoel Jerusalem F.C., isLocatedIn, Jerusalem</td>
</tr>
<tr>
<td>Robert Louis, influences, David Leavitt</td>
<td>David Louhoushon, hasGender, David Leavitt</td>
<td>Gore Vidal, influences, David Leavitt</td>
</tr>
<tr>
<td>Princess Henriette, hasChild, Violante Bavaria</td>
<td>Jonava, isAffiliatedTo, Violante Bavaria</td>
<td>Ferdinand Maria, hasChild, Violante Bavaria</td>
</tr>
</tbody>
</table>

AMLP 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 on DistMult and ConvE methods. Figure 5 shows the effect of AMLP on some of the most frequent relations. As is shown, both of the DistMult and ConvE provide a more robust representation for isAffiliatedTo and isConnectedTo relations demonstrating the confidence of models in identifying these relations. Moreover, the AMLP affects DistMult more in playsFor and isMarriedTo relations while affecting ConvE more in isConnectedTo relations.

**Examples** The adversarial attacks for a few samples are provided in Table 5. As is shown, the AMLP-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 type as their object. Moreover, we can see the most influential fact for predicting the target triples by the attack selected by AMLP-Remove. In both of the models, the relation isConnectedTo between two entities identifies by another isConnectedTo type link in the neighborhood of the original entities. Further, the models determine the relation isLocatedIn between two entities with another location type relation which identify the location of the object. For the relation wasBornIn, the most influential fact corresponds to the relation isLocatedIn for the object. Both models identify the relation hasChild through another relation hasChild for the spouse of the subject entity. We expand upon these intuitions in the next section.
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 AMLP-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 find simple patterns on subgraphs that surround the target triple and the removed fact from AMLP-Remove, and appear more than 90% of the time.

We only focus on extracting length-2 horn rules, i.e., \( R_1(a,c) \land R_2(c,b) \implies R(a,b) \), where \( R(a,b) \) is the target and \( R_2(c,b) \) is the removed fact.

Table 6 shows extracted YAGO3-10 rules that are common to both models, and ones that are not. The rules show several interesting inferences, such that \( \text{hasChild} \) is often inferred via married parents, and \( \text{isLocatedIn} \) via transitivity. There are several differences in how the models reason as well; DistMult often uses the \( \text{hasCapital} \) as an intermediate step for \( \text{isLocatedIn} \), while ConvE incorrectly uses \( \text{isNeighbor} \). We also compare against rules extracted by Yang et al. [2015a] 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 AMLP, demonstrating that AMLP 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 AMLP 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 as the incorrect triple is provided in the Figure 6. We compare our method with two baselines, 1) randomly choosing one of the neighbors as the incorrect fact, 2) choose the fact with the lowest score in the neighborhood as the incorrect triple (since we expect it to be the most incompatible with the rest of the graph). As shown, our method outperforms both of these baselines with a considerable gap. We also study the scenario where we do not assume we know that each target fact has a single error in its neighbor, instead only know about the total number of errors. We plot the number of predicted errors vs. the number of correctly detected errors by assigning a global threshold on the approximation of the change of scores for 1512 neighbors of our target samples in the Figure 7. Similar to accuracy, our method provides a more accurate detection of errors compared to the baselines.
Table 6: **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 AMLP-Remove is presented in red.

<table>
<thead>
<tr>
<th>Rule Body, $R_1(a,c) \land R_2(c,b)$</th>
<th>$\Rightarrow$</th>
<th>Target, $R(a,b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Common to both</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isConnectedTo$(a,c)$ \land isConnectedTo$(c,b)$</td>
<td></td>
<td>isConnectedTo$(a,b)$</td>
</tr>
<tr>
<td>isConnectedTo$(a,c)$ \land isConnectedIn$(c,b)$</td>
<td></td>
<td>isConnectedIn$(a,b)$</td>
</tr>
<tr>
<td>isConnectedTo$(a,c)$ \land hasChild$(c,b)$</td>
<td></td>
<td>hasChild$(a,b)$</td>
</tr>
<tr>
<td><strong>only in DistMult</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>playsFor$(a,c) \land isConnectedIn$(c,b)$</td>
<td></td>
<td>wasBornIn$(a,b)$</td>
</tr>
<tr>
<td>dealsWith$(a,c) \land participatedIn$(c,b)$</td>
<td></td>
<td>participatedIn$(a,b)$</td>
</tr>
<tr>
<td>isConnectedTo$(a,c) \land hasCapital$(c,b)$</td>
<td></td>
<td>wasBornIn$(a,b)$</td>
</tr>
<tr>
<td><strong>only in ConvE</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>influences$(a,c) \land influences(c,b)$</td>
<td></td>
<td>influences$(a,b)$</td>
</tr>
<tr>
<td>isConnectedIn$(a,c) \land hasNeighbor$(c,b)$</td>
<td></td>
<td>isConnectedIn$(a,b)$</td>
</tr>
<tr>
<td>hasCapital$(a,c) \land isConnectedIn$(c,b)$</td>
<td></td>
<td>exports$(a,b)$</td>
</tr>
<tr>
<td>hasAdvisor$(a,c) \land graduatedFrom$(c,b)$</td>
<td></td>
<td>graduatedFrom$(a,b)$</td>
</tr>
</tbody>
</table>

**Extractions from DistMult [Yang et al., 2015a]**

|                   |               |                  |
| isLocatedIn$(a,c) \land isConnectedIn$(c,b)$ |               | isConnectedIn$(a,b)$ |
| isConnectedTo$(a,c) \land isConnectedIn$(c,b)$ |               | wasBornIn$(a,b)$ |
| playsFor$(a,c) \land isConnectedIn$(c,b)$ |               | wasBornIn$(a,b)$ |
| isConnectedTo$(a,c) \land isConnectedIn$(c,b)$ |               | diedIn$(a,b)$ |

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 an 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. Although AMLP is primarily applicable to multiplicative scoring functions [Nickel et al., 2011, Socher et al., 2013, Yang et al., 2015b, Trouillon et al., 2016], these ideas are expandable to additive scoring functions [Bordes et al., 2013a, Wang et al., 2014, Lin et al., 2015, Nguyen et al., 2016] as well.

Furthermore, there is a growing body of literature that incorporates an extra type of evidence to provide more informative embeddings by combining the entity and its features representations. We can further utilize the AMLP on those that build their embeddings on top of a multiplicative scoring function. As a result, using AMLP, we can gain a deeper understanding of these methods which use an extra type of evidence 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 a combination of them [Pezeshkpour et al., 2018].

**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
Figure 6: The accuracy of detecting error in the neighborhood of 100 chosen samples. We choose the neighbor with the highest value according to Eq (??) as the incorrect fact. This experiment assumes we know each target fact has exactly one error.

Figure 7: We plot the number of predicted errors vs the number of correctly detected errors by assigning a global threshold on the approximation of the change of targets’ score.

et al., 2016, Dong et al., 2017, Zhao et al., 2018] to attain the interpretability, and further, evaluate the robustness of those models. Koh and Liang [2017] uses influence function to provide a novel approach to understand 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 which 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 the 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 adversarial training to improve their representation of the graph [Minervini et al., 2017, Cai and Wang, 2017].
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 AMLP, an adversarial modification for link prediction models: identifying the fact to add into or remove from the KG that changes the prediction for a target fact. AMLP 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 AMLP can effectively reduce ranking metrics on link prediction models upon applying the attack triples. Further, we incorporate the AMLP to study the interpretability of KG representations by summarizing the most influential facts for each relation. Finally, using AMLP, we introduce a novel automated error detection method for knowledge graphs. Code to reproduce the results will be available here: http://anon.url.
References


Appendix A. Further Proofs

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' \rangle and \langle s, r', o' \rangle. Defining the scoring function as \( \psi(s, r, o) = f(e_s, e_r) \cdot e_o = z_{s,r} \cdot e_o \), we further assume that \( \psi(s, r, o) = e_s \cdot g(e_r, e_o) = e_s \cdot x_{r,o} \).

A.1 Modifications in 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:

\[
\psi(s, r, o) - \psi(s, r, o) = (\overline{e}_s - e_s)x_{s,r} \quad (11)
\]

where \( x_{s,r} = g(e_r, e_o) \).

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

\[
\nabla_{e_s} \mathcal{L}(\overline{G}) = \nabla_{e_s} \mathcal{L}(G) - (1 - \varphi)x_{r',o'} \quad (12)
\]

where \( x_{r',o'} = g(e'_r, e'_o) \), 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:

\[
0 \simeq -(1 - \varphi)x_{r',o'}^T + (H_s + \varphi(1 - \varphi)x_{r',o'}^T x_{r',o'})(\overline{e}_s - e_s) \quad (13)
\]

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

\[
\overline{e}_s - e_s = (1 - \varphi)(H_s + \varphi(1 - \varphi)x_{r',o'}^T x_{r',o'})(\overline{e}_s - e_s) \quad (18)
\]
In practice, $H_s$ is positive definite, making $H_s + \varphi(1 - \varphi)x_{t',o}'x_{t',o}'$ positive definite as well, and invertible. Then, we compute the score change as:

$$
\nabla(s, r, o) - \psi(s, r, o) = x_{r,o}(\bar{e}_s - e_o) \\
= ((1 - \varphi)(H_s + \varphi(1 - \varphi)x_{t',o}'x_{t',o}')^{-1}x_{t',o}')x_{r,o}.
$$

### A.2 Modifications in the Form $(s, t', o)$

In this section we approximate the effect of attack in the form of $(s, t', o)$. In contrast to $(s', t', o)$ attacks, for this scenario we need to consider the change in the $e_o$ upon applying the attack, in approximation of the change in the score as well. Using previous results, we can approximate the $\bar{e}_o - e_o$ as:

$$
\bar{e}_o - e_o = (1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s,r,o}'z_{s,r,o})^{-1}z_{s,r,o}
$$

and similarly, we can approximate $\bar{e}_s - e_s$ as:

$$
\bar{e}_s - e_s = (1 - \varphi)(H_s + \varphi(1 - \varphi)x_{t',o}'x_{t',o})^{-1}x_{t',o}'
$$

where $H_s$ is the Hessian matrix over $e_s$. Then using these approximations:

$$
z_{s,r}(\bar{e}_o - e_o) = z_{s,r}(1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s,r,o}'z_{s,r,o})^{-1}z_{s,r,o}
$$

and:

$$
(\bar{e}_s - e_s)x_{r,o} = ((1 - \varphi)(H_s + \varphi(1 - \varphi)x_{t',o}'x_{t',o})^{-1}x_{t',o}')x_{r,o}
$$

and then calculate the change in the score as:

$$
\nabla(s, r, o) - \psi(s, r, o) = \\
z_{s,r}(\bar{e}_o - e_o) + (\bar{e}_s - e_s)x_{r,o} = \\
z_{s,r}(1 - \varphi)(H_o + \varphi(1 - \varphi)z_{s,r,o}'z_{s,r,o})^{-1}z_{s,r,o} \\
+ ((1 - \varphi)(H_s + \varphi(1 - \varphi)x_{t',o}'x_{t',o})^{-1}x_{t',o}')x_{r,o}
$$

### 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, $\psi(s, r, o)$ (where $\psi(s, r, o) = |e_s + e_r - e_o|$), we need to compute $\bar{e}_o$. To do so, we need to derive an efficient computation for $\bar{e}_o$. First, the derivative of the loss $\mathcal{L}(G) = \mathcal{L}(G) + \mathcal{L}((s', r', o))$ over $e_o$ is:

$$
\nabla_{e_o}\mathcal{L}(G) = \nabla_{e_o}\mathcal{L}(G) + (1 - \varphi)\frac{z_{s',r',o} - e_o}{\psi(s', r', o)}
$$

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

$$
0 \simeq (1 - \varphi)\frac{(z_{s',r',o} - e_o)\psi(s', r', o)}{\psi(s', r', o)^2} + (H_o - H_{s',r',o})(\bar{e}_o - e_o)
$$

$$
H_{s',r',o} = (1 - \varphi)\frac{(z_{s',r'} - e_o)\psi(s', r', o)}{\psi(s', r', o)^2} + \frac{1 - \varphi}{\psi(s', r', o)^2} - (1 - \varphi)\frac{(z_{s',r'} - e_o)\psi(s', r', o)}{\psi(s', r', o)^3}
$$
where $H_o$ is the $d \times d$ Hessian matrix for $o$, i.e., second order derivative of the loss w.r.t. $e_o$, computed sparsely. Solving for $e_o$ gives us:

$$e_o = -(1 - \varphi)(H_o - H_{s', r', o})^{-1} \frac{(z_{s', r'} - e_o)^\top}{\psi(s', r', o)} + e_o$$  \hspace{1cm} (21)

Then, we compute the score change as:

$$\overline{\psi}(s, r, o) = |e_s + e_r - e_o|$$  \hspace{1cm} (22)

$$= |e_s + e_r + (1 - \varphi)(H_o - H_{s', r', o})^{-1} \frac{(z_{s', r'} - e_o)^\top}{\psi(s', r', o)} - e_o|$$

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