Error Detection on Knowledge Graphs with Triple Embedding

Yezi Liu Qinggang Zhang Mengnan Du Xiao Huang† Xia Hu

***Abstract*—Knowledge graphs (KGs), as an essential ingredient in many real-world applications, always contain a considerable number of errors. KG error detection aims to find the triples whose head entity, tail entity, and corresponding relation are mismatched. Though urgently needed, existing methods are not generalizable. They mainly utilize supervised information such as entity type or erroneous labels, but such information is not often available in the real world. It remains challenging to detect errors in KGs. First, KGs have unique data characteristics compared to general graphs. Second, real-world KGs are often large, while the labels are rare and unavailable for error detection. To bridge the gap, we propose a novel KG error detection framework based on triple embedding, termed *TripleNet*. Specifically, we first construct a triple network by considering each triple as a node meanwhile connecting them with shared entities. We then exploit a Bi-LSTM layer to capture the intra-triple translational information (local level) and employ a graph attention network to gather the inter-triple contextual information (global level), respectively. Finally, we compute the triple's suspicious score by integrating its local-level and global-level information. Experimental results on two real-world KGs demonstrated that TripleNet outperforms state-of-the-art error detection algorithms with comparable or even better efficiency.**

***Index Terms*—Article submission, IEEE, IEEEtran, journal, LaTeX, paper, template, typesetting.**

I. Introduction

Knowledge graph (KG) has been demonstrated to be an efficient data structure for storing and organizing relations and knowledge during the digital age. A knowledge graph is a directed graphs in which each realistic fact has been reformulated into a triple (i.e., a head entity, a relation, and a tail entity). KGs are growing in popularity because of their flexibility in handling complex data, including web data, commercial data, general-purpose knowledge, and domain knowledge. For example, many large-scale general-purpose KGs have been created, such as Freebase [1], DBpedia [2], YAGO [3], and NELL [4]. Numerous domain-specific KGs are also emerging, such as agricultural KGs [5], biomedical KGs[1], and COVID-19 KGs[2]. These KGs have served as an essential ingredient in various artificial intelligence systems, including search engines [6], recommender systems [7], and conversational agents [8].

KG-based systems are heavily affected by KG errors. It has become infeasible to manually build and maintain real-world KGs, as they often contain millions or billions of entities [1]–[4]. Instead, heuristic automated systems have been used to extract knowledge from semi-structured or unstructured crowdsourcing sources [2], [4]. Errors were unavoidably introduced into the KGs as a result of noise in these sources and imperfections in the acquisition algorithms. For example, NELL, a KG created by a never-ending learning agent, has an estimated precision of 74% [4]. These errors would significantly affect downstream tasks, such as recommendation [8], searching [6], question answering [9], and reasoning [10]. Thus, there is an increasing demand for automatically and systematically detecting errors in KGs.

While there are extensive efforts on KG error detection, these approaches are not generalizable. Many studies take advantage of entity types to perform clustering-based outlier detection [11], [12]. However, entity types are only partially (or even not) available in real-world KGs. And another line of methods utilize path-based rule mining for error checking [13], but are limited by the coverage and quality of the rules. Several recent studies proposed to build classifiers to evaluate each triple, based on different features, including entity categories, path features, out-degrees, as well as embedding representations of entities and relations [14], [15]. However, the labels are often not available for training the classifiers. Furthermore, efforts have also been made to employ additional sources of information, such as related webpages [16] and annotations [17], to facilitate error detection. While such webpages and annotations are valuable for error detection algorithms, acquiring such supplementary information could be difficult and expensive. Therefore, these methods are not generalizable in real-world applications.

It remains a challenging task to detect errors in real-world KGs. First, KGs have unique data characteristics, including directed triples, relation with different types, and semantic properties. Second, real-world KGs are often large [2]–[4], but with rare labels. For example, Freebase contains 1.9 billion triples, and there exist more than 7000 entities and 4000 relations [1]. In this paper, we claim that detecting errors in KGs is equivalent to identifying triples that have mismatched components, i.e., the head entity, tail entity, and relation. We propose an effective solution - TripleNet. The proposed TripleNet framework firstly constructs a triple network by considering each triple as a node meanwhile connecting them with shared head or tail entity. It then uses Bi-LSTM to capture the translational structure within each triple to obtain the local representation, and an attention mechanism for collecting contextual information from neighbors to obtain the global representation. The error detection is performed based on both local dissimilarity and global inconsistency of triples. Through the investigation, we aim at answering the following

University of California Irvine. yezil3@uci.edu The Hong Kong Polytechnic University. qinggangg.zhang@connect.polyu.hk, xiaohuang@comp.polyu.edu.hk New Jersey Institute of Technology. mengnan.du@njit.edu Rice University. xia.hu@rice.edu

[1] https://bioportal.bioontology.org [2] https://covidgraph.org

two research questions. 1) How can we learn an appropriate vector space by jointly embedding translational structure information within each triple and contextual information from neighborhoods? 2) How can we conduct effective error detection in this vector space? The major contributions of our work are summarized as follows:

- We formally define the problem of error detection on knowledge graphs as a mismatch of the head entity, tail entity, and the corresponding relation.
- We propose a tailored KG error detection solution TripleNet, to detect noisy triples by considering both the local translational structure within the triple and the global contextual information from the neighborhoods of the target triple.
- Experimental results validate the effectiveness of TripleNet on two real-world knowledge graph datasets in terms of two different evaluation metrics.

II. Problem Statement

In this section, we introduce the notation as well as the problem of error detection on knowledge graphs that we target to tackle. Let $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$ denote a knowledge graph that contains a large number of triples, where $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively. Each triple is made up of a head entity h, a relation r, and a tail entity t, represented as (h, r, t). We embed the local translational structure into a basic triple embedding $\mathbf{x}$ by concatenating the bidirectional hidden state sequences output from the Bi-LSTM model. The basic embedding of the j^{th} neighbor of the anchor triple is denoted as x_j. And the final triple embedding is z. The suspicious score function is defined as $f_s(\cdot)$.

Notations: We use an uppercase bold alphabet to denote a matrix (e.g., $\mathbf{W}$) and a lowercase bold alphabet to represent a vector (e.g., $\mathbf{x}$). The transpose of a matrix is denoted as $\mathbf{W}^\top$. We use $\|\mathbf{x}\|_2$ to represent the ℓ^2 norm of a vector. The operation $\mathbf{x} = [\mathbf{h}; \mathbf{r}; \mathbf{t}]$ denotes the concatenation of the column vectors $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ into a new column vector $\mathbf{x}$. We list the major symbols in this paper in Table I.

Since the entities naturally exist in KG, we define the error in the given KG as a mismatch of the head entity, tail entity, and the corresponding relation. For example, (*Bruce_Lee*, *place_of_birth*, *Chinatown*) is a false triple, though *Bruce_Lee* and *Chinatown* are correct entities. This means that the errors are not from an individual entity and relation, but from the mismatch of three components.

Given the notations and definitions mentioned above, we formally define the problem of error detection on KGs as follows.

> Given a knowledge graph $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$, our goal is to design an end-to-end framework that takes the KG as input and returns a rank of all triples with their suspicious scores, i.e., the possibility of being an error. The performance of error detection is measured by both Precision@K and Recall@K.

III. The Proposed TripleNet Framework

In this section, we introduce the TripleNet framework. We first construct a triple network by considering each triple as a node and connecting them with shared head or tail entities. Then we apply a Bi-LSTM model to learn the translational structure within each triple and obtain basic representations of all triples. Next, we feed them into the attention layers to obtain an updated representation for the triples of the network. Finally, the suspicious score of each triple is calculated using the dissimilarity within a triple, and the inconsistency between the basic representation and the updated embedding containing neighborhood information.

TABLE I: Important symbols and definitions.

Notations	Definitions
$\mathcal{G}$	A knowledge graph
$\mathcal{T}$	A triple network reformulated from $\mathcal{G}$
$\mathcal{R}$	Relation set
$\mathcal{R}'$	Relation set in triple network
$\mathcal{E}$	Entity set
$\mathcal{S}$	Triple set
(h, r, t)	A triple of head, relation, and tail
$(\mathbf{h}, \mathbf{r}, \mathbf{t})$	Embedding of head, relation and tail
$\mathbf{x}$	Basis embedding of a triple
$\mathbf{x}_j$	Basis embedding of j^{th} neighbor
$\mathbf{z}$	Final representation of the anchor triple
$n = \|\mathcal{S}\|$	Number of triples
m	Number of neighbors of a triple
γ	Margin parameter
$f_s(\cdot)$	Suspicious score function

A. Translational-based Triple Representation

In this section, we perform translation-based triple representation learning on a KG, $\mathcal{G}$, in order to capture the local directional relations within a triple and obtain an initial representation $\mathbf{x}$ of the triple.

Given an anchor triple (h, r, t) in $\mathcal{G}$, the core idea behind the knowledge graph embedding approach is to define a mapping function $f(g(\cdot))$, where $g(\cdot) : \mathbb{R} \to \mathbb{R}^D$ is a lookup table that maps entities or relation indexes of a triple into D-dimension feature space while $f(\cdot) : \mathbb{R}^D \to \mathbb{R}^d$ is a transformation function that maps the input feature vector into a low-dimensional embedding space. The distinctions between various embedding-based methods rely on their way to determine $f(\cdot)$ as well as the loss function [18], [19]. Although it is simple and scalable, we found that it itself is not applicable to the triple-level error detection problem as shown in the ablation study, since it only measures the inner sequential error within a triple instead of the entire KG. To this end, a novel error detector should be able to directly predict the suspicious probability of a triple by leveraging its inner translational structure at the local level and the community structure of the entire triple at the global level.

To achieve the above requirements, the crucial step is to represent an individual triple as an embedding vector. That is, we aim to obtain a general representation $\mathbf{x} \in \mathbb{R}^d$ for a triple $(\mathbf{h}, \mathbf{r}, \mathbf{t})$. Basically, the naive solution is to adapt pooling methods, i.e., average and max pooling, or concatenation operation. However, they may result in suboptimal representation in practice, since they ignore the direction of knowledge graphs, e.g., the translational or sequential structure inside a triple. Therefore, we aim to explicitly capture the sequential nature of the knowledge graph and treat each triple as an ordered

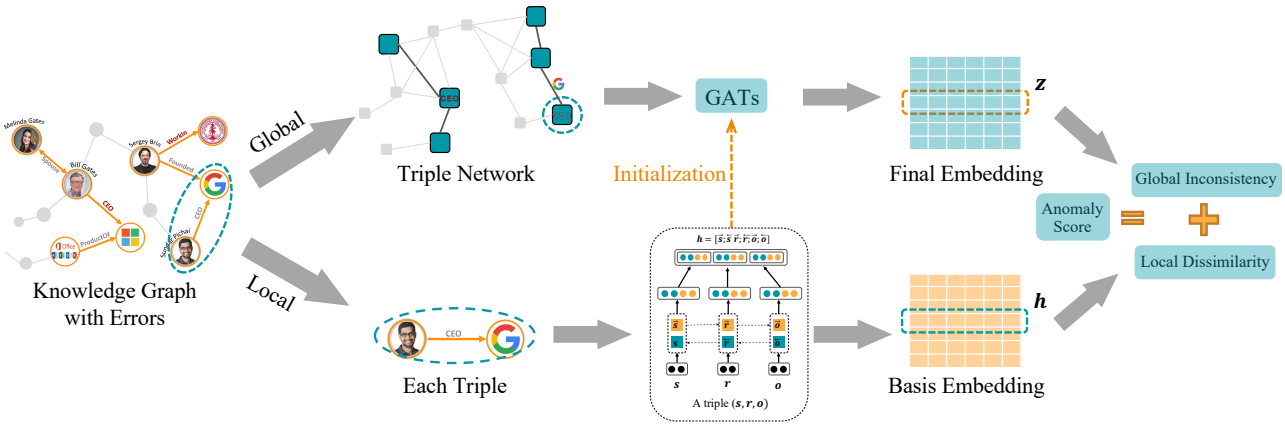

Fig. 1: The proposed TripleNet framework. TripleNet uses the Bi-LSTM to embed the local translation structure within each triple to obtain the local representation $\mathbf{x}$, and an attention mechanism to embed the triple network globally to obtain the global representation $\mathbf{z}$. The error detection is performed based on both local representation $\mathbf{x}$ and global representation $\mathbf{z}$ of triples. Eventually, a suspicious score could be obtained for each triple.

sequence from $\mathbf{head} \rightarrow \mathbf{relation} \rightarrow \mathbf{tail}$. To be specific, we implement $f(\cdot)$ as a bidirectional long short-term memory (Bi-LSTM) [20] network. Let $\mathbf{e}_T$ represent the input feature vectors, where T has three time steps, i.e., $T \in \{h, r, t\}$. Taking the backward one as an example, the hidden representation $\overleftarrow{\mathbf{e}}_T$ is computed as follows:

$$
\begin{aligned}
\mathbf{f}_T &= \sigma\left(\mathbf{W}_f \mathbf{e}_T + \mathbf{U}_f \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_f\right), \\
\mathbf{i}_T &= \sigma\left(\mathbf{W}_i \mathbf{e}_T + \mathbf{U}_i \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_i\right), \\
\mathbf{o}_T &= \sigma\left(\mathbf{W}_o \mathbf{e}_T + \mathbf{U}_o \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_o\right), \\
\mathbf{c}_T &= \mathbf{f}_T \odot \mathbf{c}_{T+1} + \mathbf{i}_T \odot \tanh\left(\mathbf{W}_c \mathbf{e}_T + \mathbf{U}_c \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_c\right), \\
\overleftarrow{\mathbf{e}}_T &= \mathbf{o}_j \odot \tanh\left(\mathbf{c}_T\right),
\end{aligned}
\tag{1}
$$

where $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_o$, $\mathbf{U}_f$, $\mathbf{U}_i$, $\mathbf{U}_o$, $\mathbf{b}_f$, $\mathbf{b}_i$, and $\mathbf{b}_o$ denote trainable parameters. $\mathbf{f}_T$, $\mathbf{i}_T$, $\mathbf{o}_T$ are respectively forget, input, and output gates at the T^{th} time stamp. $\odot$ presents the element-wise multiplication operation, while $tanh$ is the tanh activation function [21]. Assuming $\mathbf{x}_h$, $\mathbf{x}_r$, and $\mathbf{x}_t$ indicate the final output of the Bi-LSTM, we can obtain the triple representation $\mathbf{x} = [\mathbf{x}_h; \mathbf{x}_r; \mathbf{x}_t] = [\overrightarrow{\mathbf{h}}, \overleftarrow{\mathbf{h}}; \overrightarrow{\mathbf{r}}; \overleftarrow{\mathbf{r}}; \overrightarrow{\mathbf{t}}; \overleftarrow{\mathbf{t}}]$. It is worth noting that the output triple embedding $\mathbf{x}$ is supposed to capture well the translational/sequential structure of the input triple, thanks to the powerful sequential module Bi-LSTM. We will explain how to detect anomalous triples with the learned triple embedding in Section III-C.

B. Neighborhood-based Context Learning

In addition to the translational nature of the knowledge graph, the global structure/neighborhood is another vital character in understanding the connectivity relationships among triples [22], [23].Inspired by this, several efforts have been made to leverage the global structure of triples for various learning tasks in recent years, i.e., knowledge graph completion [19]. The key assumption behind these methods is that connected entities should be similar, and thus the representation of a target entity can be aggregated from its neighborhood recursively.

Motivated by the success, we aim to derive a different view of community structure by using the representations of neighbors to determine the suspicious propensity score for the anchor triple, instead of using them to learn the triple representation. To be specific, let $\mathbf{x}$ denote the embedding vector of an anchor triple as defined in Section III-A, and $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ indicates the corresponding embedding vectors of neighborhood triples, where m is the number of neighbors. Then we need to obtain the context triple embedding $\mathbf{z}$ of the anchor triple using a readout function denoted by $\mathbf{z} = \text{Readout}(\{\mathbf{x}_j\}_{j=1}^{m})$.

The readout function could be initialized in several ways in practice. We follow the work [19], [24] to implement it with attention layers due to its ability to selectively aggregate messages from different neighborhood triples. The attention mechanism can better reconstruct the anchor triple in the error detection task on KGs, as it can reduce the attention weights for noisy or abnormal neighboring triples. Specifically, when the anchor node is a positive triple, the attention mechanism could reduce information coming from anomalous neighborhoods; while the anchor node is an erroneous one, the reconstruction error between the anchor triple and its neighbors is inevitably high with equal attention weights, since the target node is inconsistent with its neighbors. In our method, we perform self-attention [24] on neighborhood triple embeddings, and the attention coefficients are computed by a shared attentional mechanism $a : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \rightarrow \mathbb{R}$ as:

$$
e_j = a(\mathbf{W}_c \mathbf{x}, \mathbf{W}_c \mathbf{x}_j), \tag{2}
$$

where e_j indicates the importance of the j^{th} neighbor to the target triple, and $\mathbf{W}_c \in \mathbb{R}^{d' \times d}$ is the weight matrix. After giving the attention coefficients, we apply a softmax function

to make them comparable as follows:

$$\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{m} exp(e_k)}, \tag{3}$$

α_j is the normalized attention score of the j-th neighbor towards the target triple. Once obtained, the context triple embedding is calculated via a non-linear combination of the neighboring triple embeddings:

$$\mathbf{z} = \sigma\left(\sum_{j=1}^{m} \alpha_j \mathbf{W}_c \mathbf{x}_j\right), \tag{4}$$

where $\sigma(\cdot)$ denotes a sigmoid activation function. In practice, when handling large-scale datasets or the number of neighbors m is large, researchers often consider a multi-head attention mechanism to improve the expressive ability or stabilize the learning process [25]. In our experiments, we found that the single-head attention has already achieved a satisfactory performance, so we do not consider the multi-head version for simplicity. Given the context embedding $\mathbf{z}$ and the corresponding anchor triple embedding $\mathbf{x}$, we argue that $\mathbf{z}$ could be used to precisely reconstruct $\mathbf{x}$, if the anchor triple is normal. This assumption is reasonable and has been widely recognized as the smoothness or clustering assumption [26] in the literature, since the normal triple lies in the same community with neighborhoods, while the malicious one is not.

C. Joint Optimization

In this section, we illustrate how to define an effective error detector for knowledge graphs based on local self-contradiction and global neighborhood inconsistency of a triple.

1) Local Dissimilarity: We define a dissimilarity function to check the self-contradiction within the triple. Many KG embedding algorithms have developed such functions to model the translational structure for better learning embeddings [18], [27], [28]. In our method, we take a simple squared Euclidean distance [18] as a dissimilarity function. To better capture the sequential nature of a triple, we utilize the hidden representation of Bi-LSTM $(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t)$, rather than the initial entities and relations embeddings $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ as in most existing methods, to explicitly compute the reconstruction loss. Specifically, the local dissimilarity is computed as follows:

$$d_{local}(\mathbf{h}, \mathbf{r}, \mathbf{t}) = ||\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t||_2. \tag{5}$$

By minimizing the above dissimilarity, the model is capable of detecting obvious abnormal triples from a local view by checking whether there is a self-contradiction in (h, r, t), in terms of semantic meaning or sequential patterns contained in the triple. However, it alone is not enough to detect KG errors effectively, since a false triple can mimic translational patterns.

2) Global Inconsistency: To address this limitation, we consider another TripleNet paradigm that judges triple anomalous based on its global structure. The basic observation is that the anomalous triple is more likely to have fake neighbors that are not related to the target triple, since the anchor triple does not exist. For this reason, we can compute the difference between an anchor triple embedding and its context embedding to determine the degree of abnormality. Formally, we calculate the difference between them using the following reconstruction error.

$$d_{global} = ||\mathbf{z} - \mathbf{x}||_2. \tag{6}$$

Compared with the local-view suspicious measurement, the above is a community-based approach and complements aligned with the local measurement. By integrating the two suspicious measurements, our model offers a comprehensive way to estimate the suspicious score of a triple as follows:

$$d_{joint} = d_{local} + \lambda d_{global}. \tag{7}$$

λ is a trade-off parameter to balance the importance of two similarities. With the joint suspicious measurement, it is helpful for the model to effectively detect errors that deviate from the community, as well as the translation nature of triples.

To encourage discrimination between positive (golden) triples and negative (corrupted) triples, we use the margin-based ranking loss function for model optimization:

$$\mathcal{L} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \left[\gamma + d_{joint}(\mathbf{h}, \mathbf{r}, \mathbf{t}) - d_{joint}(\mathbf{h}', \mathbf{r}, \mathbf{t}')\right]_+ . \tag{8}$$

where $[]_+$ denotes the positive part of x, $\gamma > 0$ is the margin hyperparameter, $\mathcal{S}$ is the set of correct triples and $\mathcal{S}'$ is the set of corrupted triples. $\mathcal{S}'$ is constructed by randomly corrupting the head or tail entity, which is defined as:

$$\mathcal{S}'_{(h,r,t)} = \{(h', r, t)|h' \in \mathcal{E}\} \cup \{(h, r, t')|t' \in \mathcal{E}\}. \tag{9}$$

D. Error Detection

After the model is properly trained, we can infer the suspicious degree of an arbitrary triple. Specifically, given a triple (h, r, t) from $\mathcal{G}$, we calculate its suspicious score $f_s(\mathbf{h}, \mathbf{r}, \mathbf{t})$ as below:

$$f_s(\mathbf{h}, \mathbf{r}, \mathbf{t}) = d_{joint}(\mathbf{h}, \mathbf{r}, \mathbf{t}). \tag{10}$$

When $f_s(\mathbf{h}, \mathbf{r}, \mathbf{t})$ is larger, the triple is more likely to be a noisy fact. In practice, to make the suspicious scores of the two terms comparable, we add a normalization layer after the Bi-LSTM network to obtain a normalized triple embedding $\mathbf{z}$. The algorithm 1 specifies the overall optimization procedure. Our framework first builds a triple network by treating each triple as a node and connecting them with sharing entities. Then it performs the triple-based representation learning from both jointly capturing the local sequential information within a triple and the global contextual information among the neighborhood triples. And finally, by checking local inconsistencies and global reconstruction errors, our framework identifies anomalous triples of the given knowledge graph in a ranking list.

IV. Experiments

We conduct extensive experiments over two real-world datasets to evaluate the effectiveness of our proposed method, and we aim to answer the following questions. **Q1.** How effective is the proposed TripleNet method compared to the state-of-the-art methods in detecting errors on KGs? **Q2.** How efficient is TripleNet compared to other anomaly detection baselines? **Q3.** How much does each component of TripleNet contribute to its prominent performance?

Algorithm 1: Proposed TripleNet framework

Input: Knowledge graph $\mathcal{G}$, triple neighbors number m, margin γ, trade-off parameter λ.
Output: Suspicious scores of input triples.

/* Triple embedding learning: */
Initialize network parameters;
Construct triple network $\mathcal{T}$;
while *not converged* **do**
  for $(h, r, t) \in \mathcal{S}$ **do**
    /* Local Triple Representation: */
    Compute the basis triple embeddings $\mathbf{x}$ using Bi-LSTM, defined in Eq.(1);
    /* Global Triple Representation: */
    Obtain the contextal embeddings of triples based on neighboring triples, defined in Eq.(2)-(4);
    /* Joint Optimization: */
    Estimate the joint reconstruction loss from both the translational structure and community aspects, defined in Eq.(7);
    Update the model parameters with stochastic gradient descent by minimizing Eq.(8).
/* Error Detection Process: */
Input the triple (h, r, t) into the TripleNet model and calculate the suspicious score as defined in Eq.(10);

TABLE II: Statistics of the knowledge graph datasets.

	# Triples (n)	# Entity	# Relation
NELL	231,634	46,682	821
DBpedia	2,920,168	976,404	504

A. Datasets

To verify the effectiveness of TripleNet, we conduct experiments over two benchmark knowledge graph datasets, NELL [4] and DBpedia [29], which are publicly accessible and vary in source domains, size, and sparsity. The statistical information of the datasets is summarized in Table II.

B. Baseline Methods

To comprehensively evaluate the performance of the proposed method, we compare it with five state-of-the-art algorithms, including TransE [18], ComplEx [30], DistMult [31], KGIST [32], and KGTtm [33]. Specifically, the first three are typical knowledge graph representation methods, including translation-based and bilinear-mapping models. The other two are state-of-the-art error detection baselines in knowledge graphs, which aim to detect abnormal triples.

C. Experimental Settings

In this section, we introduce the overall experimental setups of this work, including dataset processing, evaluation metrics, and parameter settings.

1) Data pre-processing.: We resort to injecting synthetic errors to stimulate the real-world erroneous environment, since there are no labeled errors in our two KG datasets. According to Section **??**, the entity naturally exist in real-word KGs. Thus a mismatched triple could either be a non-related pair of entities with a link, or a triple that contains a false relation. Following the previous research [33], we inject these two types of errors into our datasets: false relation error and connection error.

The idea of generating a false relation error is to swap the relation of a golden standard triple with a false one. To make sure the generated one is false, we first randomly select a triple from the dataset and perturb it by replacing the relation while keeping its head and tail entities. We resort to producing the connection error is to insert a false link between a non-related pair of entities. To get a non-related pair, we randomly select two entities and a relation in our dataset and connect them into a triple, while making sure that the generated triple is not in the original dataset. Note here we use index i and j here to indicate the relation and that the tail does not necessarily come from the same triple.

To demonstrate that our method could perform effective and stable error detection on KGs, which have different portions of errors, we set up the error ratio p from $\{1\%, 2\%, 3\%, 4\%, 5\%\}$. We uniformly and randomly inject a mix of false relation errors and connection errors, with a ratio $p/2\%$ correspondingly, so that each sample triple has an equal probability of being picked up to construct an incorrect triple. Therefore, in our experiments, the goal is to automatically detect these perturbed triples.

2) Evaluation Metrics: The model ranks all the triples in the dataset according to their predicted suspicious scores to evaluate the performance of our method with comparing to all the baselines. A larger score will result in a higher rank, which indicates a higher probability of being an error. Based on the ranking, we further evaluate our error detection results by calculating the two evaluation measures defined as follows:
Precision@K evaluates the proportion of true errors that we discovered in the top K triples.
Recall@K measures the proportion of true errors that we discovered in the total number of ground truth errors.

$$\text{Precision@K} = \frac{\mid \text{Errors Discovered} \mid \bigcap \mid \text{Top K in Ranking} \mid}{\mid \text{Top K in Ranking} \mid}$$

$$\text{Recall@K} = \frac{\mid \text{Errors Discovered} \mid \cap \mid \text{Top K in Ranking} \mid}{\mid \text{Total Errors} \mid}$$

3) Parameter Settings: We use the codes released or provided of the baselines to conduct experiments. We implement our TripleNet in PyTorch. The embedding size is fixed to 100 for all models. We optimize all models with Adam optimizer, where the batch size is fixed at 1024. We use the default Xavier initializer to initialize our model parameters, and the initial learning rate is 0.001. For TripleNet_Local, TripleNet_Global, TripleNet_GAT, we set the hidden dimension of Bi-LSTM layers to 100. The number of neighbors is computed by taking the average number of neighbors of all triples in a dataset. By making this setting, we make sure that our model could adapt to datasets with different levels of density of neighbors, in order to best utilize the neighborhood information. For our two datasets, NELL and DBpedia, the corresponding numbers of neighbors are set to 5, and 3, which are the average number of neighbors for all triples in the two datasets, respectively. For each sample in the synthetic dataset, we treat it as a positive instance and follow the negative sampling approach in [33] to construct negative samples. Since we may inject different errors in different runs, to reduce randomness, we use the random seed and report the average results of ten runs.

TABLE III: Error detection results of Precision@K and Recall@K based on two datasets with error ratio $p = 5\%$.

Precision@K

Dataset	NELL					DBpedia				
Top@K	1%	2%	3%	4%	5%	1%	2%	3%	4%	5%
TransE	0.637	0.531	0.427	0.412	0.366	0.645	0.548	0.476	0.423	0.383
ComplEx	0.601	0.526	0.454	0.419	0.348	0.603	0.533	0.472	0.431	0.357
DistMult	0.631	0.532	0.472	0.423	0.401	0.662	0.539	0.489	0.438	0.420
KGIST	0.675	0.586	0.496	0.459	0.431	0.701	0.613	0.501	0.498	0.450
KGTtm	0.681	0.600	0.512	0.452	0.405	0.760	0.628	0.586	0.474	0.436
Ours	**0.738**	**0.623**	**0.538**	**0.477**	**0.435**	**0.844**	**0.729**	**0.632**	**0.557**	**0.497**

Recall@K

Dataset	NELL					DBpedia				
Top@K	1%	2%	3%	4%	5%	1%	2%	3%	4%	5%
TransE	0.127	0.212	0.256	0.330	0.366	0.129	0.219	0.286	0.338	0.383
ComplEx	0.120	0.210	0.272	0.335	0.348	0.121	0.213	0.283	0.345	0.357
DistMult	0.126	0.213	0.283	0.338	0.401	0.132	0.216	0.293	0.350	0.420
KGIST	0.134	0.234	0.298	0.367	0.431	0.140	0.245	0.301	0.398	0.450
KGTtm	0.136	0.240	0.307	0.362	0.405	0.152	0.251	0.352	0.379	0.436
Ours	**0.148**	**0.249**	**0.323**	**0.382**	**0.436**	**0.169**	**0.292**	**0.379**	**0.445**	**0.497**

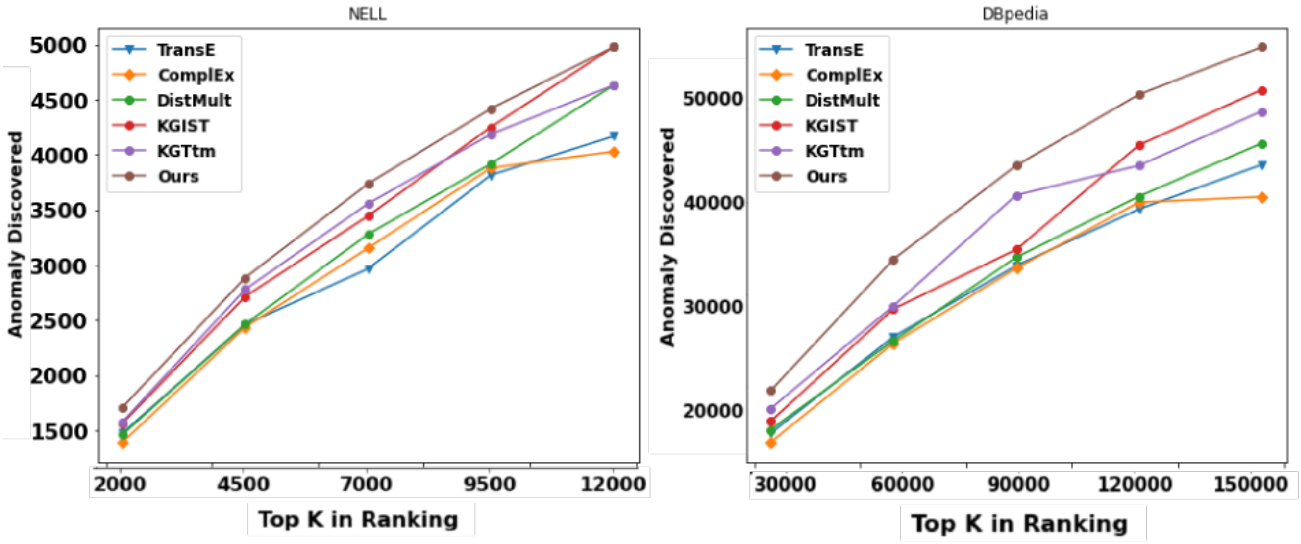

Fig. 2: Anomaly Discovery Curve.

TABLE IV: The running time for each iteration (in seconds).

	TransE	ComplEx	DistMult	KGIST	KGTtm	Ours
NELL	1	1	40	52	4	1
DBpedia	20	21	96	122	33	38

D. Effectiveness of TripleNet (Q1)

We first evaluate the effectiveness of our TripleNet model over the baselines on two benchmark datasets. Table III and Figure 2 report the results of comparing methods for error detection on two metrics, i.e., Precision@K and Recall@K. We have the following three observations.

- From Table III, we can observe that TripleNet consistently performs better than other baselines over all the datasets with a great margin. In particular, TripleNet improves over the second-best results by 5% at $K = 1\%$, and 10% at $K = 2\%$ in NELL and DBpedia datasets, respectively. In general, compared to error detection methods, knowledge graph representation baselines such as TransE, ComplEx, and DistMult yield worse results in general. The reason is that the general KG representation learning frameworks do not consider the errors in KG, thus do not learn discriminative representations for normal and noisy triples. This result demonstrates the need to develop task-specific algorithms for error detection.
- The previous error detection methods (KGIST and KGTtm) are always surpassed by our model across two datasets, in terms of two evaluation metrics. This is mainly because our model is capable of exploiting the sequential pattern within the triples, as well as their global structure for error detection.

TABLE V: Ablation Study on NELL with 5% ratio of errors.

	Top@K	1%	2%	3%	4%	5%
$Precision@K$	Triple_Local	0.674	0.571	0.497	0.446	0.406
	Triple_Global	0.714	0.619	0.526	0.464	0.422
	Triple_GAT	0.738	0.623	0.538	0.477	0.435
	Top@K	1%	2%	3%	4%	5%
$Recall@K$	Triple_Local	0.135	0.228	0.291	0.357	0.406
	Triple_Global	0.143	0.247	0.315	0.371	0.422
	Triple_GAT	0.148	0.249	0.323	0.382	0.436

The other two methods could only use the learned rules or sampled paths in the detection. Both rule-based or path-based methods are coarser than our fine-granularity detector based on individual triple and its neighborhoods. Note that KGIST utilizes extra label information for training, but we only use self-contained information of KGs. These results demonstrate the effectiveness of our model in capturing local and global information for error detection in KGs.

- From Table III and Figure 2, we can see that the proposed model consistently outperforms other baselines across different K values, in terms of Precision@K and Recall@K. And with the increasing number of K, the performance gap between our method and other baselines tends to increase. It is because other baselines could only detect the errors with obvious patterns and thus, noisy triples may escape the detection. We contribute this success of our method to the joint optimization of individual triple-level and global-level error detectors. This result further validates the effectiveness of our model for error detection.

E. Efficiency Analysis of TripleNet (Q2)

To answer **Q2**, we record the running time of one iteration for all models. All experiments are conducted with one NVIDIA RTX 2080 Ti GPU. From Table IV, we can observe that TransE and ComplEx run faster than other methods in general, since they only need to calculate the mean square losses. The semantic-based method DistMult costs more time than TransE and ComplEx on all datasets. Compared to error detection methods, our model is comparable to the path-based error detection method KGTtm on average, while it runs faster than the rule-based method, KGIST, especially in the large-scale dataset, DBpedia. This observation validates the efficiency of our model compared to baseline methods.

F. Ablation Study (Q3)

To test the effect of the components in the proposed method, we carry out an ablation study. Specifically, we introduce three variants. TripleNet_Local, TripleNet_Global to validate the effectiveness of modeling translational patterns based on the Bi-LSTM layer, for the community-based suspicious measurement, and self-attention network for context embedding aggregation, respectively. TripleNet_Local only considers the local dissimilarity measurement defined in Eq. (5), which is different from TransE because it includes the Bi-LSTM layer to explicitly model the sequential patterns. TripleNet_Global only considers the global dissimilarity measurement defined in Eq.(6). Note that TripleNet_GAT indicates our final version that adopts the attention mechanism to implement the Readout function. Table V summarizes the results on two datasets with 5% errors.

We have the following observations according to the results. First, TripleNet_Local performs better than TransE, but worse than TripleNet_GAT. It validates the effectiveness of the Bi-LSTM layer in capturing sequential patterns within the triples for better representation learning. Second, TripleNet_Global performs better than TripleNet_Local in most cases. It indicates that the majority of anomalous triples could be better detected by checking the global structure among the triples, while some are deceptive and can be detected by checking their local sequential structure. Third, TripleNet_GAT outperforms both TripleNet_Local and TripleNet_Global with a great margin. This result indicates the complementary effects of local and global measurements for joint anomaly detection. It validates our motivation to jointly consider the translational nature within triples and their global connectivity for a more accurate error detection. It demonstrates the effectiveness of the attention network in selectively aggregating neighborhood triple information for target triple reconstruction.

V. RELATED WORK

In this section, we discuss two types of KG error detection methods that are most relevant to ours, including embedding-based and rule-mining-based error detection methods. We then contextualize our work in the literature and distinguish our task from other KG tasks, such as KG completion and KG link prediction.

A. Knowledge Graph Error Detection

KG *error detection* is one of the two subtasks of KG refinement, the goal of which is to make a KG more comprehensive and accurate. KG error detection aims at removing erroneous information from an existing KG, while the other subtask, KG *completion*, aims at adding more knowledge. We will discuss KG completion in the later subsection. The two tasks are strictly distinct, and generally, there are no approaches that could do both [34]. Existing KG error detection methods can be further categorized into embedding-based methods or rule-mining-based methods.

B. Embedding-based Error Detection

KG embedding-based error detection methods generally follow two steps. First, they use some KG representation algorithms to map entities and relations to a lower-dimensional space. Then they use the obtained embedding vectors to define an anomaly score or confidence score for each triple to measure the suspiciousness of entities, relations, or triples.

To obtain lower dimensional embeddings, existing methods include tensor factorization-based models [35], [36], translational distance models such as TransE [18], and semantic

matching models such as ComplEx [30]. The main family is the translational distance models [37]. The first work of this family is TransE, which reflects the idea that the embedding vector of the subject plus the embedding vector of the relation should be close to the vector of the object for a given triple, meaning $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. TransH [38] and TransR [39] expand upon TransE, a model's relation-specific approach, by modeling a relation as a translating operation on a hyperplane or even modeling relations and entities in two distinct spaces. This allows the two models to deal with 1-to-N, N-to-1, and N-to-N relations that TransE cannot manage. TransM [40] tries to solve this problem by relaxing the Translation Requirement mentioned above.

There are also other embedding-based methods. For instance, the Triple trustworthiness measurement model for knowledge graph (KGTtm) [41] uses a crisscrossed neural network-based structure, combining different elements through a multi-layer perceptron fusioner to generate confidence scores for each triple. Besides, ComplEx [30] first employs complex vector space and uses Hermitian dot product to do relation composition for a head entity, relation, and the conjugate of a tail entity.

C. Rule-mining-based Error Detection

In data mining, association rule mining is a kind of method that analyzes the co-occurrence of items in itemsets and leverages these association rules for error detection.

All of these approaches are based on association rule mining [42], which is a simple but effective method for error detection. AMIE [43] introduces an altered confidence metric based on the partial completeness assumption, which applies a particular relationship of an entity to obtain all relationships of that type for that entity. The following work of AMIE named AMIE+ [44] adapts to larger knowledge graphs and takes examples of complete and incomplete assertions as training data, and predicts completeness of predicate types observe during the training process. [45] proposed a Graph-Repairing Rules named GRRs [45], aiming to identify incomplete, conflicting, and redundant information in graphs and indicated how to correct these errors. [46] propose to use (in-)completeness meta-information to better assess the quality of rules learned from incomplete KGs. RUGE [47] enables an embedding model to learn simultaneously from labeled triples, unlabeled triples, and soft rules in an iterative manner. KALE [48] was proposed to jointly embed knowledge graphs and logical rules, the key idea of which is to represent and model triples and rules in a unified framework.

D. Link Prediction

The goal of KG link prediction is to evaluate the trustworthiness of the relations between entity pairs, and it is different from the error detection task because the former evaluates the connectivity of an entity pair, but the latter one further considers if an entity pair mismatches the corresponding relation.

E. KG Completion

KG completion intends to increase the coverage of KGs [34] by predicting missing information, such as entities, types of entities, and relations. Predicting the types of entities mainly utilize some classification methods that are trained on labeled data. In [49], [50], they treat the KG completion problem as predicting the conditional probability of an entity give other entities, and if they are connected, the probability would be high.

In summary, although these tasks have strictly different goals, they share some common techniques in solving the problem. For example, they all used embedding-based or rule-mining-based methods as the solutions. However, to achieve better performance on each task, the representation and rule mining algorithms should be tailored for the specific task. And that is why we need to propose a novel tailored KG embedding method for the error detection task.

VI. Conclusion and Future Work

In this paper, we investigate the error detection problem based on triple-level embedding. We propose a novel error detection framework, termed *TripleNet*. Our general idea is to utilize KG self-information to calculate local inconsistency, which uses triples' self-contained sequential information; and global mismatch, which uses KG's contextual information to reconstruct a triple given a KG, and form them into the suspicious score. Extensive experiments on two real-world knowledge graph datasets demonstrate the rationality and effectiveness of TripleNet. In the future, we would like to explore the use of the embedding method in TripleNet to guide KG reasoning and question answering. More sophisticated and advanced models [27], [28] will also be explored further in our future research.

REFERENCES

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008, pp. 1247–1250.

[2] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic Web journal*, vol. 6, no. 2, pp. 167–195, 2015.

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW*, 2007, pp. 697–706.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI*, 2010.

[5] Y. Chen, J. Kuang, D. Cheng, J. Zheng, M. Gao, and A. Zhou, "Agrikg: an agricultural knowledge graph and its applications," in *DASFAA*, 2019, pp. 533–537.

[6] J. S. Eder, "Knowledge graph based search system," Jun. 21 2012, uS Patent App. 13/404,109.

[7] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *WWW*, 2019, pp. 3307–3313.

[8] K. Zhou, W. X. Zhao, S. Bian, Y. Zhou, J.-R. Wen, and J. Yu, "Improving conversational recommender systems via knowledge graph based semantic fusion," in *KDD*, 2020, pp. 1006–1014.

[9] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," *WSDM*, pp. 105–113, 2019.

[10] A. Polleres, A. Hogan, A. Harth, and S. Decker, "Can we ever catch up with the web?" *Semantic Web journal*, vol. 1, pp. 45–52, 2010.

[11] J. Debattista, C. Lange, and S. Auer, "A preliminary investigation towards improving linked data quality using distance-based outlier detection," in *JIST*, 2016, pp. 116–124.

[12] H. Paulheim and A. Gangemi, "Serving DBpedia with DOLCE–more than just adding a cherry on top," in *ISWC*, 2015, pp. 180–196.

[13] B. Shi and T. Weninger, "Discriminative predicate path mining for fact checking in knowledge graphs," *Knowledge-based Systems*, vol. 104, pp. 123–133, 2016.

[14] S. Jia, Y. Xiang, X. Chen, and E. Shijia, "TTMF: A triple trustworthiness measurement frame for knowledge graphs," *Computing Research Repository*, 2018.

[15] A. Melo and H. Paulheim, "Detection of relation assertion errors in knowledge graphs," in *K-CAP*, 2017.

[16] J. Lehmann, D. Gerber, M. Morsey, and A.-C. N. Ngomo, "Defacto-deep fact validation," in *ISWC*, 2012, pp. 312–327.

[17] M. N. Jeyaraj, S. Perera, M. Jayasinghe, and N. Jihan, "Probabilistic error detection model for knowledge graph refinement," in *CICLing*, 2019.

[18] A. Bordes, N. Usunier, A. Garcia-Duran *et al.*, "Translating embeddings for modeling multi-relational data," in *NeurIPS*, 2013.

[19] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, "Learning attention-based embeddings for relation prediction in knowledge graphs," *arXiv preprint arXiv:1906.01195*, 2019.

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] E. Fan, "Extended tanh-function method and its applications to nonlinear equations," *Physics Letters A*, vol. 277, no. 4-5, pp. 212–218, 2000.

[22] Z. Sun, C. Wang, W. Hu *et al.*, "Knowledge graph alignment network with gated multi-hop neighborhood aggregation," in *AAAI*, 2020.

[23] B. Oh *et al.*, "Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods," in *CIKM*, 2018.

[24] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[26] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, "Focused clustering and outlier detection in large attributed graphs," in *KDD*, 2014.

[27] D. Nathani, J. Chauhan, C. Sharma *et al.*, "Learning attention-based embeddings for relation prediction in knowledge graphs," in *ACL*, 2019.

[28] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2018.

[29] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[30] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction." ICML, 2016.

[31] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[32] C. Belth, X. Zheng, J. Vreeken, and D. Koutra, "What is normal, what is strange, and what is missing in a knowledge graph: Unified characterization via inductive summarization," in *WWW*, 2020, pp. 1115–1126.

[33] S. Jia, Y. Xiang, X. Chen, and K. Wang, "Triple trustworthiness measurement for knowledge graph," in *WWW*, 2019, pp. 2865–2871.

[34] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

[35] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in *Icml*, vol. 11, 2011, pp. 809–816.

[36] ——, "Factorizing yago: scalable machine learning for linked data," in *WWW*, 2012, pp. 271–280.

[37] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *TKDE*, vol. 29, no. 12, pp. 2724–2743, 2017.

[38] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Twenty-Eighth AAAI*, 2014.

[39] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[40] M. Fan, Q. Zhou, E. Chang, and F. Zheng, "Transition-based knowledge graph embedding with relational mapping properties," in *PACLIC*, 2014.

[41] S. Jia, Y. Xiang, X. Chen, and E. Shijia, "Ttmf: A triple trustworthiness measurement frame for knowledge graphs." *CoRR*, 2018.

[42] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 SIGMOD*, 1993.

[43] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, "Amie: association rule mining under incomplete evidence in ontological knowledge bases," in *WWW*, 2013, pp. 413–422.

[44] L. Galárraga, C. Teflioudi, K. Hose *et al.*, "Fast rule mining in ontological knowledge bases with amie," *The VLDB Journal*, 2015.

[45] Y. Cheng, L. Chen, Y. Yuan, and G. Wang, "Rule-based graph repairing: Semantic and efficient repairing methods," in *ICDE*. IEEE, 2018, pp. 773–784.

[46] T. P. Tanon, D. Stepanova, S. Razniewski, P. Mirza, and G. Weikum, "Completeness-aware rule learning from knowledge graphs," in *ISWC*. Springer, 2017, pp. 507–525.

[47] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Knowledge graph embedding with iterative guidance from soft rules," in *AAAI*, 2018.

[48] ——, "Jointly embedding knowledge graphs and logical rules," in *EMNLP*, 2016, pp. 192–202.

[49] H. Paulheim and C. Bizer, "Type inference on noisy rdf data," in *International semantic web conference*. Springer, 2013, pp. 510–525.

[50] ——, "Improving the quality of linked data using statistical distributions," *IJSWIS*, vol. 10, no. 2, pp. 63–86, 2014.