ERROR DETECTION IN KNOWLEDGE GRAPHS

A Thesis

by

YEZI LIU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2021

Major Subject: Computer Science

ABSTRACT

Knowledge graphs (KGs) are an efficient tool for storing and organizing information. While KGs are increasingly used as an essential ingredient in a wide range of applications, they always contain a considerable number of errors, especially for newly emerging KGs. These errors were inevitably introduced when extracting KGs from crowdsourcing websites by using heuristic programs. KG error detection aims to find the triples whose head entity, tail entity, and the corresponding relation are mismatched. Though it is increasingly urgent, existing KG error detection algorithms are not generalizable. It remains a challenge to detect errors in KGs. First, KGs have unique data characteristics comparing to general graphs. Second, real-world KGs are often large, while the labels are rare and not available for error detection. To bridge the gap, we propose a novel KG error detection framework based on triple embedding, termed as *TripleNet*. 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 to capture the local-level translational information within the triple, and employ an attention mechanism to gather the contextual information on the global level. Finally, we define the triple suspicious score by integrating the local dissimilarity and the global inconsistency between the basic embedding and the embedding after fusing information from neighbor triples. Experimental results on two real-world KGs demonstrated that TripleNet outperforms state-of-the-art error detection algorithms, with comparable or even better efficiency.

ACKNOWLEDGMENTS

The past two years in Texas A&M University is a unforgettable journey in my life. I have received a great deal of support and assistance. Here I would like to thank all the people who have helped me in my research.

First and foremost, I would like to thank my advisor Dr. Xia (Ben) Hu, for his help, encouragement, and guidance.

Second, I would like to thank my committee members, Professor Anxiao Jiang and Professor Xiaoning Qian for their guidance and support throughout the course, and their kindly help.

Also, I would like to thank my group members and collaborators at the DATA (Data Analytics at Texas A&M) Lab in the Department of Computer Science and Engineering, for helping me and providing advice for my work.

I owe my sincerest thank my parents for their support and love.

Finally, I would like to thank the Texas A&M University and the funding agencies for supporting my research during my master's program.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by the thesis committee consisting of Professor Xia (Ben) Hu [advisor], Professor Anxiao Jiang from the Department of Computer Science and Engineering and Professor Xiaoning Qian from the Department of Electrical and Computer Engineering.

All the work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University and DARPA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background and Motivation

Knowledge graphs (KGs) have been demonstrated to be an efficient data structure for storing and organizing relations and knowledge during the digital age. They are 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 since 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 intelligent systems, including search engines [6], recommender systems [7], and conversational agents [8].

KG-based systems are heavily influenced by errors in KGs. It has become infeasible to manually build and maintain real-world KGs, as they often contain millions or billions of entities [1, 2, 3, 4]. Instead, heuristic automated systems have been used to extract knowledge from semi-structured or unstructured crowdsourcing sources [2, 4]. Due to the noises in these sources and the imperfection of acquisition algorithms, errors were inevitably introduced into KGs. For instance, NELL, a KG created by a never-ending learning agent, has an estimated precision of 74% [4]. YAGO3 is a large-scale KG with over 150 million facts [3], GeoNames [3]. and the accuracy of YAGO3 is about 95%, corresponding to around 7.5 million errors [3]. Wikidata has been frequently vandalized as anyone could edit it [9]. These errors would significantly affect downstream tasks, such

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

as recommendation [8], searching [6], question answering [10], and reasoning [11]. Thus, there is an increasing demand for automatically and systematically detecting errors in the 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 [12, 13]. 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 [14], but they 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 [15, 16, 17]. However, the labels are often not available for training the classifiers. Besides, efforts also have been devoted to employing extra information sources, such as related webpages [18] and annotations [19], to facilitate error detection. While such webpages and annotations are valuable for the error detection algorithms, acquiring such supplemental information could be hard and expensive. Therefore, these methods are not generalizable in real-world applications.

It remains a challenging task to detect errors on 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, 3, 4], but with rare labels. For instance, 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.

1.2 Thesis Contributions

Through the investigation, we aim at answering the following 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.

1.3 Related Work

In this section, we discuss two kinds 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.

1.3.1 Knowledge Graph Error Detection

KG *error detection* is one of the two subtasks of KG refinement, whose goal 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*, targets 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[20]. Existing KG error detection methods can be further categorized into embedding-based methods, or rule-mining-based methods.

1.3.1.1 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 [21, 22], translational distance models such as TransE [23], and semantic matching models such as ComplEx [24]. The main family is the translational distance models [25]. 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 [26] and TransR [27] expand upon TransE, a model's relation-specific approach, by modeling a relation as a translating operation on a hyper-plane or even modeling relations and entities in two distinct spaces. This allows for the two models to deal with 1-to-N, N-to-1, and N-to-N relations that TransE cannot manage. TransM [28] tries to tackle 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) [29] 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 [24] 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.

1.3.1.2 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.

These approaches are all based on association rule mining [30], which is a simple but effective method for error detection. AMIE [31] 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+ [32] 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. [33] proposed a Graph-Repairing Rules named GRRs [33], aiming to identify incomplete, conflicting, and redundant information in graphs and indicated how to correct these errors. [34] propose to use (in-)completeness meta-information to better assess the quality of rules learned from incomplete KGs. RUGE [35] enables an embedding model to learn simultaneously from labeled triples, unlabeled triples, and soft rules in an iterative manner. KALE [36] 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.

1.3.2 Other Tasks in Knowledge Graphs

1.3.2.1 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.

1.3.2.2 KG Completion

KG completion intends to increase the coverage of KGs [20] by predicting missing information, such as entities, types of entities, and relations. Predicting the types of entities mainly utilize some classification methods training on labeled data. In [37, 38], 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.

To sum up, 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.

2. PROBLEM STATEMENT

In this section, we introduce the notations as well as the the problem of error detection on knowledge graphs that we target to tackle.

Notations: We use an uppercase bold alphabet to denote a matrix (e.g., **W**) and a lowercase bold alphabet to represent a vector (e.g., **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 concatenating column vectors **h**, **r**, and **t** into a new column vector **x**.

We list the major symbols in this paper in Table 2.1. 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 composed 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 **x** by concatenating the bidirectional hidden state sequences output from 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 $f_s(\cdot)$.

Since the entities naturally exist in KG, we define the error on 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 aforementioned notations and definitions, we formally define the problem of error detection on KGs as follows.

Table 2.1: 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 of the target triple
$\mathbf{z}$	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

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

3. THE PROPOSED TRIPLENET FRAMEWORK

In this section, we introduce TripleNet framework. We firstly 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 in order to obtain an updated representation for the nodes/triples of the network. Finally, the suspicious score of each triple is calculated based on the dissimilarity inside a triple, and the inconsistency between the basic representation and the updated embedding containing neighborhood information.

3.1 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 knowledge graph embedding approach is to define a mapping function $f(g(\cdot))$, where $g(\cdot) : R \to R^D$ is a lookup table that maps entities or relation indexes of a triple into D-dimension feature space while $f(\cdot) : R^D \to R^d$ is a transformation function that maps the input feature vector into 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 [23, 39]. For example, TransE parameterizes $f(\cdot)$ as a simple identity function and trains the model by minimizing a translation-based loss function, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. After training, the loss score could be used to determine the suspiciousness of a triple. Although it is simple and scalable, we found that it alone is not applicable for the triple-level error detection problem as shown in our experiments, 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

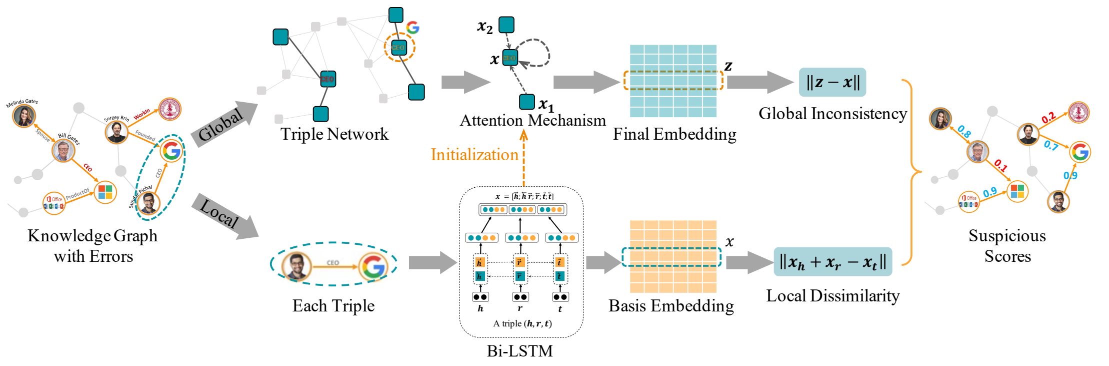

Figure 3.1: The proposed TripleNet framework. TripleNet uses the Bi-LSTM to embed the local translation structure within each triple to obtain the local representation **x**, and an attention mechanism to embed the triple network globally to obtain the global representation **z**. The error detection is performed based on both local representation **x** and global representation **z** of triples. Eventually, a suspicious score could be obtained for each triple.

capable of directly predicting the suspicious probability of a triple by leveraging its inner translational structure at the local level as well as 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 $\mathbf{x} \in R^d$ 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 knowledge graph and treat each triple as a ordered sequence from **head** $\rightarrow$ **relation** $\rightarrow$ **tail**. To be specific, we implement $f(\cdot)$ as a bidirectional long short-term memory (Bi-LSTM)[40] 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:

$$\mathbf{f}_T = \sigma(\mathbf{W}_f \mathbf{e}_T + \mathbf{U}_f \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_f), \tag{3.1}$$

$$\mathbf{i}_T = \sigma(\mathbf{W}_i \mathbf{e}_T + \mathbf{U}_i \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_i), \tag{3.2}$$

$$\mathbf{o}_T = \sigma(\mathbf{W}_o \mathbf{e}_T + \mathbf{U}_o \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_o), \tag{3.3}$$

$$\mathbf{c}_T = \mathbf{f}_T \odot \mathbf{c}_{T+1} + \mathbf{i}_T \odot \tanh(\mathbf{W}_c \mathbf{e}_T + \mathbf{U}_c \overleftarrow{\mathbf{e}}_{T+1} + \mathbf{b}_c), \tag{3.4}$$

$$\overleftarrow{\mathbf{e}}_T = \mathbf{o}_j \odot \tanh(\mathbf{c}_T), \tag{3.5}$$

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 [41]. Assume $\mathbf{x}_h$, $\mathbf{x}_r$, and $\mathbf{x}_t$ indicate the final output of the Bi-LSTM, then 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 to note that the output triple embedding $\mathbf{x}$ is supposed to well capture 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 3.3.

3.2 Neighborhood-based Context Representation

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 [42, 43]. 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 [39]. The key assumption behind these methods is that the 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 target 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 **x** denote the embedding vector of an anchor triple as defined in Section 3.1, and $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$ indicates the corresponding embedding vectors of neighborhood triples, in which m is the number of neighbors. Then we need to obtain the context triple embedding **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 [39, 44] 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 that comes 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 [44] on neighborhood triple embeddings, and the attention coefficients are computed by a shared attentional mechanism $a : R^{d'} \times R^{d'} \to R$ as

$$e_j = a(\mathbf{W}_c\mathbf{x}, \mathbf{W}_c\mathbf{x}_j). \tag{3.6}$$

Where e_j indicates the importance of j^{th} neighbor to the target triple, and $\mathbf{W}_c \in R^{d' \times d}$ is the weight matrix. We then normalize the coefficients across all m neighborhood triples so that to make the embeddings **z** and **x** comparable:

$$\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{m} exp(e_k)}. \tag{3.7}$$

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:

$$\mathbf{z} = \sigma\left(\sum_{j=1}^{m} \alpha_j \mathbf{W}_c \mathbf{x}_j\right), \tag{3.8}$$

where $\sigma(\cdot)$ denotes a sigmoid activation function. In practice, when handling large-scale datasets or the number of neighbors m is big, researchers often consider a multi-head attention mechanism to improve the expressive ability or stabilize the learning process [45]. 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 **z** and the corresponding anchor triple embedding **x**, we argue that **z** could be used to precisely reconstruct **x**, if the anchor triple is normal. This assumption is reasonable and has been widely recognized as the smoothness or clustering assumption [46] in the literature, since the normal triple lies in the same community with neighborhoods while the malicious one is not.

3.3 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.

3.3.1 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 [23, 47, 48]. In our method, we take a simple squared euclidean distance [23] 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 entity and relation embeddings **(h,r,t)** as in most existing methods, to explicitly compute the reconstruction loss. Specifically, the local dissimilarity is computed as follows.

$$d_{local}(\textbf{h,r,t}) = ||\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t||_2. \tag{3.9}$$

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 that contains in the triple. However, it alone is not enough to detect the errors on KGs effectively, since a false triple may mimic the translational patterns.

3.3.2 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 actually 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.

$$d_{global} = ||\mathbf{z} - \mathbf{x}||_2. \tag{3.10}$$

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.

$$d_{joint} = d_{local} + \lambda d_{global}. \tag{3.11}$$

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. And since d_{local} and d_{global} are different in scales, we first normalize them into range 0 to 1 and then integrate them in the one joint function.

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

$$\mathcal{L} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \left[\gamma + d_{joint}(\mathbf{h},\mathbf{r},\mathbf{t}) - d_{joint}(\mathbf{h}',\mathbf{r},\mathbf{t}')\right]_+ . \tag{3.12}$$

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

$$\mathcal{S}'_{(h,r,t)} = \{(h',r,t)|h' \in \mathcal{E}\} \cup \{(h,r,t')|t' \in \mathcal{E}\}. \tag{3.13}$$

3.4 Error Detection

After the model is properly trained, we can inference 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.

$$f_s(\mathbf{h},\mathbf{r},\mathbf{t}) = d_{joint}(\mathbf{h},\mathbf{r},\mathbf{t}). \tag{3.14}$$

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 get normalized triple embedding **z**. 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 the local inconsistency and global reconstruction errors, our framework identifies the anomalous triples of the given knowledge graph in a ranking list.

Algorithm 1 The proposed TripleNet framework

```
Input: Knowledge graph $\mathcal{G}$, number of triple neighbors $m$, margin $\gamma$, trade-off parameter $\lambda$.
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 x using Bi-LSTM, defined in Eq. (3.1)-(3.5)
            /* Global Triple Representation:                         */
        Obtain the contextal embeddings of triples based on neighboring triples, defined in
            Eq.(3.6)-(3.8) /* Joint Optimization:                    */
        Estimate the joint reconstruction loss from both the translational structure and commu-
          nity aspects, defined in Eq.(3.11)  Update the model parameters with stochastic gradient
          descent by minimizing Eq.(3.12).
    end
end
/* Error Detection Process:                                          */
Input the triple $(h, r, t)$ into the TripleNet model and calculate the suspicious score as defined in
  Eq.(3.14)
```

4. EXPERIMENTS

We conduct extensive experiments over two real-world datasets to evaluate the effectiveness of our proposed method. Specifically, we aim to answer the following questions.

- **Q1.** How effective is the proposed method TripleNet compared with the state-of-the-art methods in detecting errors on KGs?
- **Q2.** How efficient is TripleNet compared with other anomaly detection baselines?
- **Q3.** How much does each component of TripleNet contribute to its prominent performance?

4.1 Datasets

To verify the effectiveness of TripleNet, we conduct experiments over two benchmark knowledge graph datasets that are publicly accessible and vary in source domains, size, and sparsity. The statistical information of the datasets is summarized in Table 4.1.

- **NELL** [4]: It refers to *Never Ending Language Learning*, which is extracted from the unstructured online corpus. Its content is exploited by continuously learning text patterns of type and relation assertions and then applying them to discovering new entities and relations. In our version, there are totally $1,115$ iterations.
- **DBpedia** [49]: It is a popular knowledge graph benchmark extracted from Wikipedia. This data is obtained by mapping key-value pairs of infoboxes to DBpedia's ontology via a crowdsourcing process [20].

Table 4.1: Statistics of the knowledge graph datasets.

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

4.2 Baseline Methods

To comprehensively evaluate the performance of the proposed method, we compare it with five state-of-the-art algorithms, including TransE, ComplEx, DistMult, KGIST, and KGTtm. Specifically, the first three are typical knowledge graph representation methods, including both 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. Note that existing error detection methods in knowledge graphs focus on using either rule or path similarities to detect errors, while our model is a general embedding-based framework, which is both effective and scalable, as shown in the latter section.

- **TransE** [23]: It is a canonical translational distance-based KG embedding approach. The basic idea is the relationship between two entities corresponding to a translation between the embeddings of entities. It assumes that the embeddings of entities and relations are in the same space, i.e., R^k.
- **ComplEx** [24]: This is a semantic matching model that utilizes complex-valued embeddings and Hermitian dot products to preserve antisymmetric relations. It represents a triple score as the Hermitian dot product of the relation, head entity, and tail entity embeddings, which consist of real and imaginary vector components.
- **DistMult** [50]: It learns embeddings from a bilinear objective, which can capture compositional semantics of relations and extract Horn rules that involve compositional reasoning.

- **KGIST** [51]: It is a rule-mining based method that uses an unsupervised KG summarization way to find erroneous, incomplete, and legitimate information on a KG. KGIST formulates the problem of finding erroneous, incomplete. It also uses prior information in implementing their method except for the KG data.

- **KGTtm** [52]: This is an embedding-based triple trustworthiness method that uses the triple semantic information and the global inference information of the knowledge graph.

4.3 Experimental Settings

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

4.3.1 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 [52], 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 to be false, we first randomly select a triple from the dataset and perturb it by replacing the relation while keeping its head and tail entities. For example, a sample type error for a triple (h, r, t) is (h, r', t), where $r' \in \mathcal{R}$, $(h, r', t) \notin \mathcal{S}$. We finally make sure the newly generated triple is false, which means that it does not exist in our original dataset.

The idea of 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 the generated triple is not in the original dataset. In other words, we connect the entities (h_i, t_j) selected from $\mathcal{E}$, and $r \in \mathcal{R}$, to create a false triple $(h_i, r, t_j) \notin \mathcal{S}$. Note here we use index i and j here to indicate the relation and the tail do not necessary come from the same triple.

To demonstrate 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 mixed of false relation errors and connection errors, with a ratio $p/2\%$ correspondingly, so that each sample triple has an equal probability to be picked up to construct an incorrect triple. Therefore, in our experiments, the goal is to automatically detect these perturbed triples.

4.3.2 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.

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

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

4.3.3 Parameter Settings

We use the released or provided codes of 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 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, correspondingly. We search optimal hyperparameters with gird search, where the learning rate is searched from $0.05, 0.01, 0.005, 0.001$, the margin parameter γ from 0 to 1, and the trade-off parameter λ is tuned between 0.001 to 1000. For each sample in the synthetic dataset, we treat it as a positive instance and follow the negative sampling approach in [52] to construct negative samples. Since we may inject different errors in different runs, to reduce the randomness, we use the random seed and report the average results of ten runs.

4.4 Effectiveness of TripleNet (Q1)

We first evaluate the effectiveness of our model TripleNet over the baselines on two benchmark datasets. We tune the hyperparameters and observe that the error detection performance are stable basically without great fluctuation as γ varies, so we set the γ to 1 in obtaining the following results. And for λ, the performance varies regarding different datasets. For NELL, the detection performance is much better when the value of λ is larger than 1, with an optimal result at 10. We figure out that the possible reason for this observation is that NELL has a more dense data structure compared with DBpedia. Therefore, there exist more triple-level neighbors in such a dense data structure, which assists our model in learning more comprehensive global triple embedding representations. In addition, DBpedia achieves optimal results when λ is around 0.1. DBpedia seems to be not sensitive with parameter λ, and its value is stable basically without great fluctuation.

Table 4.2 and Figure 4.1 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 for our experimental results.

Table 4.2: 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**

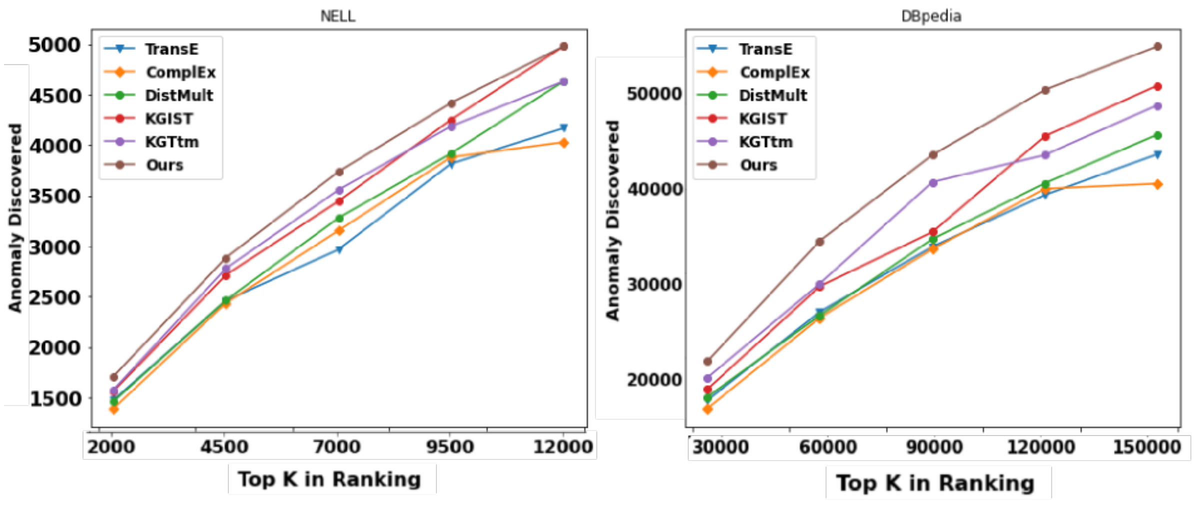

Figure 4.1: Anomaly Discovery Curve.

- From Table 4.2, 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, comparing with 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 necessity to build task-specific algorithms for error detection.

- Former 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. The other two methods could only use the learned rules or sampled paths in 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 4.2 and Figure 4.1, 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 gap of the performances 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.

4.5 Efficiency Analysis of TripleNet (Q2)

In this section, we aim at answering the second question. We record the average computation time of one iteration for all models, and summarize the results in Table 4.3. All experiments are conducted with one NVIDIA RTX 2080 Ti GPU.

From Table 4.3, we can observe that TransE and ComplEx run faster than other methods in general since they only need to calculate the mean square losses. And the semantic-based method DistMult costs more time than TransE and ComplEx on all datasets. Comparing 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 comparing to baseline methods. In addition, considering the fact that our model not only runs comparably or even faster than baseline methods in both small and large datasets, but also achieves significantly better performance than them across two evalua-

Table 4.3: The computation time for each iteration (in seconds) of different methods.

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

tion metrics. In summary, our model is more efficient for the anomaly detection task on knowledge graphs.

Table 4.4: Error detection results on the proposed method and two variants, in terms of 5% ratio of errors.

Precision@K					
Top@K	1%	2%	3%	4%	5%
Triple_Local	0.6736	0.5707	0.4971	0.4460	0.4055
Triple_Global	0.7142	0.6185	0.5256	0.4637	0.4215
Triple_GAT	0.7383	0.6231	0.5375	0.4768	0.4354
Recall@K					
Top@K	1%	2%	3%	4%	5%
Triple_Local	0.1347	0.2282	0.2912	0.3567	0.4055
Triple_Global	0.1428	0.2474	0.3153	0.3710	0.4215
Triple_GAT	0.1476	0.2492	0.3225	0.3815	0.4354

4.6 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. (3.9), which is different from TransE because it includes Bi-LSTM layer to explicitly model the sequential patterns. TripleNet_Global only considers the global dissimilarity measurement defined in Eq.(3.10). Note that TripleNet_GAT indicates our final version that adopts the attention mechanism to implement the Readout function. Table 4.4 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 a reconstruction of the target triple.

5. CONCLUSION AND FUTURE WORK

5.1 Conclusion

Knowledge graphs are a crucial tool that contains authoritative knowledge. Automatically detecting errors in KGs is essential and promising for dynamically updating, large-scare KGs. Advances in graph neural networks motivate us to explore triple-embedding-based error detection on KGs. However, it is a challenging problem since KGs contain highly heterogeneous knowledge that is hard to properly represent this knowledge in a format that a machine could use. In this paper, we investigate the error detection problem based on triple-level embedding. We propose a novel error detection framework, termed as *TripleNet*. We make the most use of the KG self-information, including triples' self-contained sequential information and KG's contextual information to reconstruct a triple given a KG. A triple is detected as a mismatch according to this combined representation information. Extensive experiments on two real-world knowledge graph datasets demonstrate the rationality and effectiveness of TripleNet.

5.2 Future Work

5.2.1 Short-Term Plan

Contrastive Knowledge Graph Error Detection Since the labels are not often available, to perform unsupervised error detection, one possible solution is to employ contrastive learning. Its core idea is to construct different views of the original data and train the models based on the comparisons between the learning on the different views [53, 54]. As a complement to the negative samples, contrastive learning could help the model learn distinguishable across-view representations. Thus, we propose to incorporate contrastive learning and negative sampling, aiming to perform effective KG error detection. And the

TripleNet could help us to construct a different view of the original KG.

Knowledge Graph Error Detection Using Semantic Information Since knowledge graphs relate to rich semantic information, like the entity description, and the text information. These features are beneficial for error detection. We could use some natural language models to take these features into consideration for KG error detection.

5.2.2 Long-Term Plan

In the future, we would like to explore using the embedding method in TripleNet for guiding KG reasoning, question answering. More sophisticated and advanced models [48, 47] would 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 *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 1247–1250, 2008.

[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.-e. M. Suchanek, G. Kasneci, and G. Weikum, "YAGO: A core of semantic knowledge," in *WWW*, pp. 697–706, 2007.

[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*, pp. 533–537, 2019.

[6] J. S. Eder, "Knowledge graph based search system," June 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*, pp. 3307–3313, 2019.

[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*, pp. 1006–1014, 2020.

[9] S. Heindorf, M. Potthast, B. Stein, and G. Engels, "Vandalism detection in wikidata," in *CIKM*, pp. 327–336, 2016.

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

[11] 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.

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

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

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

[15] C. Ge, Y. Gao, H. Weng, C. Zhang, X. Miao, and B. Zheng, "Kgclean: An embedding powered knowledge graph cleaning framework," *arXiv preprint arXiv:2004.14478*, 2020.

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

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

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

[19] M. N. Jeyaraj, S. Perera, *et al.*, "Probabilistic error detection model for knowledge graph refinement," in *CICLing*, 2019.

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

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

[22] M. Nickel, V. Tresp, and H.-P. Kriegel, “Factorizing yago: scalable machine learning for linked data,” in *WWW*, pp. 271–280, 2012.

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

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

[25] Q. Wang, Z. Mao, B. Wang, and L. Guo, “Knowledge graph embedding: A survey of approaches and applications,” *TKDE*, 2017.

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

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

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

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

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

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

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

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

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

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

[36] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Jointly embedding knowledge graphs and logical rules," in *EMNLP*, pp. 192–202, 2016.

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

[38] H. Paulheim and C. Bizer, "Improving the quality of linked data using statistical distributions," *IJSWIS*, vol. 10, no. 2, pp. 63–86, 2014.

[39] 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.

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

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

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

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

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

[45] 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*, pp. 5998–6008, 2017.

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

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

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

[49] 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*, pp. 722–735, Springer, 2007.

[50] 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.

[51] 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*, pp. 1115–1126, 2020.

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

[53] K. Hassani and A. H. K. Ahmadi, “Contrastive multi-view representation learning on graphs,” in *ICML*, 2020.

[54] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, “Graph representation learning via graphical mutual information maximization,” in *WWW*, 2020.

APPENDIX A

CODES

Codes of our method: https://github.com/TripleNetCodes/TripleNet.git

Codes for the baseline methods:

TransE https://github.com/datquocnguyen/STransE

ComplEx https://github.com/ttrouill/complex

DistMult https://github.com/mana-ysh/knowledge-graph-embeddings

KGIST https://github.com/GemsLab/KGist

KGTtm https://github.com/TJUNLP/TTMF