Error Detection on Knowledge Graphs with Triple Embedding

Yezi Liu[1] Qinggang Zhang[2] Mengnan Du[3] Xiao Huang[2] Xia Hu[4]
[1]University of California Irvine [2]The Hong Kong Polytechnic University
[3]New Jersey Institute of Technology [4]Rice University
yezil3@uci.edu {qinggangg.zhang, xiaohuang}@comp.polyu.edu.hk
mengnan.du@njit.edu xia.hu@rice.edu

***Abstract*—Knowledge graphs (KGs), as essential ingredients 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. Despite being urgently needed, existing KG error detection methods lack generalizability. They mainly utilize supervised information such as entity type or erroneous labels, but such information is not often available in the real world. The challenges in detecting errors in KGs are twofold. Firstly, KGs exhibit unique data characteristics that distinguish them from general graphs. Secondly, real-world KGs tend to be large, and labels are often scarce. To bridge the gap, we propose a novel KG error detection framework based on *triple embedding*, termed TripleNet. TripleNet constructs a triple network by treating each triple as a node and connecting them via shared entities. It then employs a Bi-LSTM module to capture intra-triple translational information at the local level and uses a graph attention network to gather inter-triple contextual information at the global level. Finally, it computes the suspicious score of each triple by integrating its local 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.**

I. Introduction

Knowledge graphs (KGs) have been regarded as an efficient data structure for storing and organizing relations and knowledge in many applications, including search engines [1], recommender systems [2], and conversational agents [3]. KGs are directed graphs, in which each real fact has been reformulated into a triple (i.e., a head entity, a relation, and a tail entity). Due to the noises in crowdsourcing sources and the imperfection of acquisition algorithms, errors were inevitably introduced when constructing KGs. For instance, NELL has an estimated precision of 74% [4], and YAGO3 has an accuracy of 95% [5]. Thus, it becomes increasingly crucial to automatically and systematically detect errors in the KGs [6].

While there are extensive efforts on KG error detection, these approaches mainly use supervised information. Specifically, existing approaches can be mainly categorized into three types: 1) *Entity-type-based*: these studies take advantage of entity types to perform clustering-based outlier detection [7], [8]. However, entity types are only partially (or even not) available in real-world KGs. 2) *Rule-based*: these studies detect errors by checking if a triple satisfies the pre-defined rules [9], so they are limited by the coverage and quality of the rules. 3) *Embedding-based*: some studies use this approach 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 [10]–[12]. However, labels are often not available for training those classifiers. Therefore, these previous methods are not generalizable in real-world applications.

Detecting errors in real-world KGs remains a challenging task. On the one hand, KGs possess unique data characteristics, such as directivity, diverse types of relations, and semantic properties, which make their representation difficult. On the other hand, real-world KGs are often large [4], [5], [13], but with rare labels. To address these challenges, we claim that detecting errors in KGs is equivalent to identifying triples that have mismatched components and propose an effective unsupervised solution - TripleNet. TripleNet constructs a Triple Network, treating each triple as a node and establishing connections among nodes through shared entities. This network facilitates the integration of global information by considering the inter-dependencies among triples. Then, it applies an attention mechanism to the Triple Network to collect the contextual information of the target triple from its neighbors. Finally, unsupervised error detection is performed based on local dissimilarity and global inconsistency of triples.

We aim to answer three research questions: (Q1) How effective is the proposed TripleNet compared with the state-of-the-art methods in detecting errors on KGs? (Q2) How efficient is TripleNet compared with anomaly detection baselines? (Q3) How much does each component of TripleNet contribute to its performance? Our main contributions are as follows:

- We present a novel global view of KGs, i.e., Triple Network, which helps represent the contextual information of triples.
- We propose an error detection framework on KGs, TripleNet, which employs a local dissimilarity and global inconsistency.
- Experimental results validate the effectiveness of TripleNet compared with state-of-the-art KG error detection methods.

II. Problem Statement

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

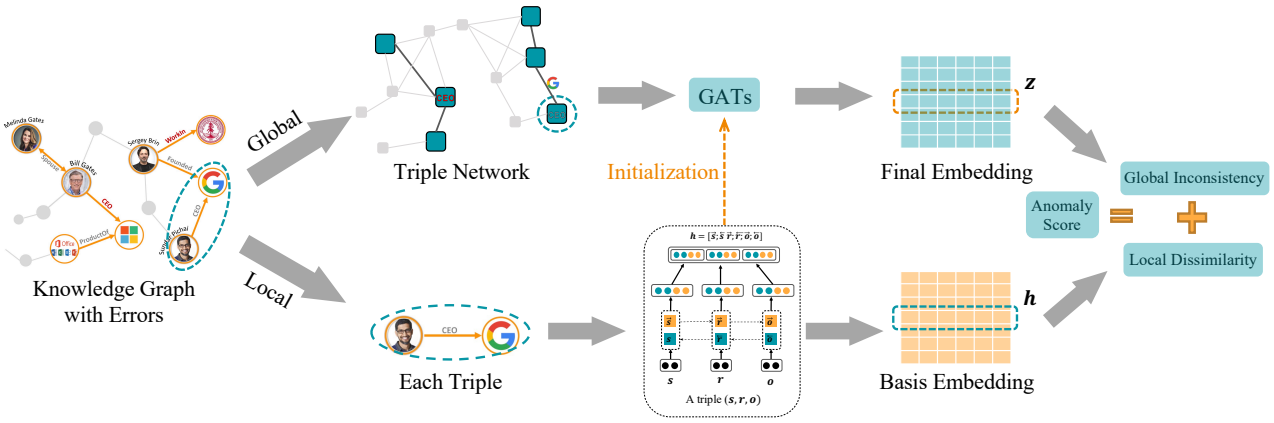

Fig. 1: The proposed TripleNet framework. TripleNet uses a Bi-LSTM layer to embed the local translation structure within each triple to obtain the local representation **x**, and an attention mechanism to embed the triple network globally to obtain the global representation **z**. The error detection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

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 concatenating column vectors $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ into a new column vector $\mathbf{x}$.

Let $\mathcal{G} = \{\mathcal{E}, \mathcal{R}\}$ denote a knowledge graph that contains n 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 construct a triple network $\mathcal{T}$, by treating each triple in $\mathcal{G}$ as a node in $\mathcal{T}$, through the connections of sharing (head or tail) entity. The relation set in $\mathcal{T}$ can be represented as $\mathcal{R}'$. We embed the local translational structure into a basic triple embedding $\mathbf{x}$ by concatenating the bidirectional hidden state sequences output from a 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 entities naturally exist in KGs, we define the error on a KG as a mismatch of the head entity, tail entity, and the corresponding relation. For example, (*Elon_Musk*, *CEO*, *Paypal*) is a false triple, though *Elon_Musk* and *Paypal* are correct entities. This implies that the errors do not originate from a singular entity or relation but from the mismatch of three components. Given the above notations and definitions, we formally define the problem of error detection on KGs as:

> 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

We elaborate our proposed framework in Figure 1. The Triple Network is constructed to better represent the contextual information of triples in a KG. A triple in the original KG is regarded as a node in the Triple Network, and a connection exists between two nodes if there is a shared (head or tail) entity between the corresponding two triples. The following subsection elaborates TripleNet model for error detection.

A. Translational-based Triple Representation

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 $f(g(\cdot))$, that maps the input feature vector into low-dimensional embedding space. The distinctions between various embedding-based methods rely on their way of determining $f(\cdot)$ as well as the loss function [14], [15]. This method is not applicable enough since it only measures the inner sequential error within a triple instead of the entire KG. A naive solution is to adapt pooling methods or concatenation operations. 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 employ a bidirectional long short-term memory (Bi-LSTM) [16] network. 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 $\mathbf{x} = [\mathbf{x}_h; \mathbf{x}_r; \mathbf{x}_t]$. We notice that in our experiment, the output triple embedding $\mathbf{x}$ can well capture the translational/sequential structure of the input triple, thanks to the powerful sequential module Bi-LSTM.

B. Contextual Aggregation via Triple Network

The global structure/neighborhood is another vital character in understanding the connectivity relationships among triples [17], [18], which has been leveraged by several efforts in recent years [15]. The key assumption behind these methods is that the connected triples that share the same entity are always semantically relevant, and thus the representation of a target triple can be aggregated from its neighborhood recursively. Motivated by the success of previous efforts [15], we derive the global inconsistency to determine the suspicious propensity score for the anchor triple. To be specific, let $\mathbf{x}$ denote an embedding vector of an anchor triple in the Triple Network,

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 $\mathbf{z}$ of the anchor triple using a readout function denoted by $\mathbf{z} = \text{readout}(\{\mathbf{x}_j\}_{j=1}^{m})$.

We follow the work [15], [19] to implement the readout function with attention layers due to its ability to selectively aggregate messages from neighborhood triples. To better reduce the weights for abnormal neighboring triples, we perform self-attention [19] on neighborhood triple embeddings, and the attention coefficients are computed by a shared attentional mechanism $a : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}$ as: $e_j = a(\mathbf{W}_c\mathbf{x}, \mathbf{W}_c\mathbf{x}_j)$, where e_j indicates the importance of j^{th} neighbor to the target triple, and $\mathbf{W}_c \in \mathbb{R}^{d' \times d}$ is the weight matrix. After that, we apply a softmax function to make attention coefficients comparable: $\alpha_j = \frac{exp(e_j)}{\sum_{k=1}^{m} exp(e_k)}$, where 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:

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

where $\sigma(\cdot)$ denotes a sigmoid activation function. 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.

C. Joint Optimization and Error Detection

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 [14], [20], [21]. In our method, we take a simple squared Euclidean distance [14] 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 **(h,r,t)** as in most existing methods, to explicitly compute the reconstruction loss. Specifically, the local dissimilarity is computed as follows:

$$d_{local}(\mathbf{h},\ \mathbf{r},\ \mathbf{t}) = ||\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t||_2, \tag{2}$$

However, checking the self-contradiction alone is not enough to detect the errors on KGs effectively, since a false triple may mimic the translational patterns.

2) Global Inconsistency: To complement the local dissimilarity, we also consider another TripleNet paradigm that judges triple anomalous based on its global structure. The basic observation is that the abnormal 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,

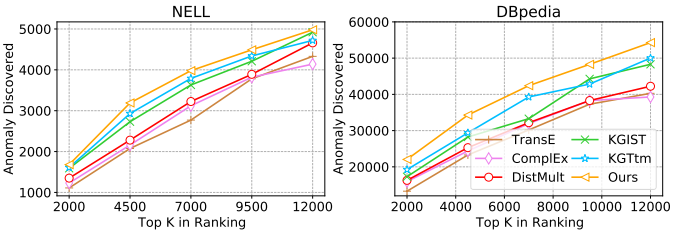

Fig. 2: Anomaly Discovery Curve on NELL and DBpedia. The x-axis is the top K triples in the anomaly score ranking, and the y-axis is the erroneous triples discovered at different K.

we calculate the difference between them using the following reconstruction error:

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

Our model further integrates the two suspicious measurements to make a complementary suspicious score as follows:

$$d_{joint} = d_{local} + \lambda d_{global},$$

where λ is a trade-off parameter to balance the importance of two similarities.

To encourage discrimination between positive (golden) triples and negative (corrupted) triples, we use an unsupervised training approach for model optimization. Specifically, the margin-based ranking loss function is defined as follows:

$$\mathcal{L} = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'} \left[\gamma + d_{joint}(\mathbf{h},\mathbf{r},\mathbf{t}) - d_{joint}(\mathbf{h}',\mathbf{r},\mathbf{t}')\right]_+, \tag{3}$$

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:

$$\mathcal{S}'_{(h,r,t)} = \{(h', r, t)|h' \in \mathcal{E}\} \cup \{(h, r, t')|t' \in \mathcal{E}\}. \tag{4}$$

3) 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 $f_s(\mathbf{h,r,t})$ as below:

$$f_s(\mathbf{h,r,t}) = d_{joint}(\mathbf{h,r,t}). \tag{5}$$

When $f_s(\mathbf{h,r,t})$ is larger, the triple is more likely to be a noisy fact. Our framework finally identifies the anomalous triples of the given KG by ranking the suspicious scores.

IV. Experiments

A. Experimental Settings

1) Dataset: To verify the effectiveness of TripleNet, we conduct experiments over two benchmark knowledge graph datasets, NELL [4] and DBpedia [22], that are publicly accessible and vary in source domains, size, and sparsity. The statistical information of the datasets is summarized in Table I.

TABLE I: Statistics of the knowledge graph datasets.

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

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

Metric	Precision@K										Recall@K									
Dataset	NELL					DBpedia					NELL					DBpedia				
Top@K	1%	2%	3%	4%	5%	1%	2%	3%	4%	5%	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	0.127	0.212	0.256	0.330	0.366	0.129	0.219	0.286	0.338	0.383
ComplEx	0.601	0.526	0.454	0.419	0.348	0.603	0.533	0.472	0.431	0.357	0.120	0.210	0.272	0.335	0.348	0.121	0.213	0.283	0.345	0.357
DistMult	0.631	0.532	0.472	0.423	0.401	0.662	0.539	0.489	0.438	0.420	0.126	0.213	0.283	0.338	0.401	0.132	0.216	0.293	0.350	0.420
KGIST	0.675	0.586	0.496	0.459	0.431	0.701	0.613	0.501	0.498	0.450	0.134	0.234	0.298	0.367	0.431	0.140	0.245	0.301	0.398	0.450
KGTtm	0.681	0.600	0.512	0.452	0.405	0.760	0.628	0.586	0.474	0.436	0.136	0.240	0.307	0.362	0.405	0.152	0.251	0.352	0.379	0.436
Ours	**0.738**	**0.623**	**0.538**	**0.477**	**0.435**	**0.844**	**0.729**	**0.632**	**0.557**	**0.497**	**0.148**	**0.249**	**0.323**	**0.382**	**0.436**	**0.169**	**0.292**	**0.379**	**0.445**	**0.497**

2) Baseline Methods: To comprehensively evaluate the performance of the proposed method, we compare it with five state-of-the-art algorithms, including TransE [14], ComplEx [23], DistMult [24], KGIST [25], and KGTtm [26]. 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.

3) Data pre-processing.: Since there are no labeled errors in our two KG datasets, we resort to injecting two types of synthetic errors to stimulate the erroneous environment in the real world: false relation error, which is to swap the relation of a golden-standard triple with a false one; and connection error, which is to insert a false link between a non-related pair of entities. We set up the error ratio p from $\{1\%, 2\%, 3\%, 4\%, 5\%\}$, and 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.

B. Effectiveness of TripleNet (Q1)

We first evaluate the effectiveness of our model TripleNet over these baselines on two benchmark datasets. Table II 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 II, we can observe that TripleNet consistently performs better than other baselines over all the datasets with a great margin. Compared with error detection methods, knowledge graph representation baselines such as TransE, ComplEx, and DistMult yield worse results in general. This result demonstrates the necessity of building 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. These results demonstrate the effectiveness of our model in capturing local and global information for error detection in KGs.
- From Table II and Figure 2, we observe 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 in the performances between our method and baselines tends to increase. The success of our method is attributed to the joint optimization of local-level and global-level error detectors.

TABLE III: The running time for one iteration (in seconds).

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

C. 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 III, 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. 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.

D. Ablation Study (Q3)

To address **Q3**, we conducted an ablation study, which involved the proposed method and its two variants: 1) TripleNet_Local, focuses solely on the local dissimilarity measurement defined in Eq. (2). 2) TripleNet_Global, considers only the global dissimilarity measurement defined in Eq. (III-C2). 3) TripleNet_GAT, denotes the proposed method in this experiment, which incorporates the attention mechanism to implement the readout function. Table IV provides a summary of the results obtained on NELL with a 5% error rate. The following observations were made: First, the performance of TripleNet_Local surpassed that of TransE but fell short of TripleNet_GAT. This outcome validates the effectiveness of the Bi-LSTM layer in capturing sequential patterns within the triples, thereby enhancing representation learning. Second, TripleNet_Global outperformed TripleNet_Local in the majority of cases. This finding suggests that the global structure among the triples holds valuable information for detecting the most anomalous triples. However, it should be noted that some anomalous triples exhibit deceptive characteristics and can be detected by examining their local sequential structure. Third, TripleNet_GAT demonstrated a significant performance improvement over both TripleNet_Local and TripleNet_Global. This outcome underscores the complementary effects of combining local and global measurements for joint error detection.

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

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

V. 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. KG embedding-based error detection methods include tensor factorization-based models [27], [28], translational distance models such as TransE [14], TransM [29], TransR [30], and TransH [31], semantic matching models such as ComplEx [23], triple trustworthiness measurement model KGTtm [32]. Rule-mining-based error detection approaches associate rule mining which analyzes the co-occurrence of items in item sets and leverages these association rules for error detection [33]–[36]. However, to achieve better performance, the representation and rule mining algorithms should be tailored for the specific task. Therefore, we need to propose a novel tailored KG embedding method for the error detection task.

VI. Conclusion and future work

Automatically detecting errors in KGs is essential and promising for dynamically updated and large-scale KGs. In this paper, we investigate the error detection problem based on triple-level embedding. We propose a novel error detection framework, termed TripleNet. We make the most use of the KG self-contained information, including triples' 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 effectiveness of TripleNet for detecting errors on real-world KGs. In the future, we would like to explore using the embedding method in TripleNet for guiding KG reasoning, and question-answering.

References

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

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

[3] 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.

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

[6] Q. Zhang, J. Dong, K. Duan, X. Huang, Y. Liu, and L. Xu, "Contrastive knowledge graph error detection," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 2590–2599.

[7] 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.

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

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

[10] 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.

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

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

[13] 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*, vol. 6, no. 2, pp. 167–195, 2015.

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

[15] 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.

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

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

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

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

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

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

[22] 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.

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

[24] 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.

[25] 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.

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

[27] 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.

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

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

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

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

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

[33] 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.

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

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

[36] 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.